Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 14, 2017

G. Huston
APNIC
P. Koch
DENIC eG
A. Durand
ICANN
W. Kumari
Google

September 10, 2016

Problem Statement for the Reservation of Special-Use Domain Names using RFC6761
draft-adpkja-dnsop-special-names-problem-06

Abstract

The dominant protocol for name resolution on the Internet is the Domain Name System (DNS). However, other protocols exist that are fundamentally different from the DNS, and may or may not share the same namespace.

When an end-user triggers resolution of a name on a system that supports multiple, different protocols or resolution mechanisms, it is desirable that the protocol used is unambiguous, and that requests intended for one protocol are not inadvertently answered using another protocol.

RFC 6761 introduced a framework by which a particular domain name could be acknowledged as being special. Various challenges have become apparent with this application of the guidance provided in RFC 6761. This document focuses solely on documenting the specific challenges created by RFC 6761 in the form of a problem statement in order to facilitate further discussions of potential solutions. In particular, it refrains from proposing or promoting any solution. Also, the current document does not focus on other general issues related to the use of special use domain names.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction: DNS, Name Space or Name Spaces, Name Resolution Protocols

For a very long time, "DNS" and "the name space" have been perceived as the same thing. However, this has not always been the case; in the past, other name resolution protocols (such as NIS, NIS+, host files, UUCP addresses, and others) were popular. Most of those have been obsoleted by the DNS in the late 1990s. More information on the history of names and namespaces can be found in [I-D.lewis-domain-names].

More recently, new name resolution protocols have been proposed, each addressing a particular need or a particular community. For example, the DONA handle system [DONA] has been used by parts of the publication industry. The Apple "Bonjour" set of protocols, inspired by what was available on Appletalk networks, was developed to perform automatic name resolution on a local IP network. The TOR project is using the onion system to obfuscate communications, the GNU Name System (GNS) system is using block chains to build a decentralized name system to offer "privacy and censorship resistance". Many more name resolution protocols have been proposed.

These alternate name resolution protocols do not exist in a vacuum. Application developers have expressed a strong desire to build their software to function in any of those universes with minimal changes. In order to do so, the software has to deterministically recognize what kind of name it is dealing with and associate it with the corresponding name resolution protocol. An algorithmic solution frequently chosen by application developers consists simply in using a special tag padded at the end of a name to indicate an alternate name resolution method. For example, if a name ends in .local, the software uses the Apple Bonjour protocol based on multicast DNS; if the name ends in .onion, it uses the TOR protocol; if the name ends in .gnu, it uses the GNS protocol, and so on. One noteworthy exception to this approach is the DONA system that has its own interoperability mechanism with the DNS. Another noteworthy exception is the Frogans technology [FROGANS] which name space uses the character '*' to separate network names from site names and allow any character, including dots on either side of the '*'.

A result of the above is that a number of applications have been developed (and massively distributed) that have encoded their favorite "tag" as a DNS TLD in a free-for-all, beginning their existence by squatting on that DNS space; .local, .gnu, .onion started out like that.

2. IETF RFC6761 Special Names

The IETF used a provision from the IETF/ICANN MoU [RFC2860] section 4.3 that says that "(a) assignments of domain names for technical uses" is to be considered the purview of IETF (outside of the scope of ICANN) in order to create a way to reserve such names in a list of "special names". That process is documented in [RFC6761] (which, however, does not directly refer the IETF/ICANN MoU). The [RFC6761] process was first applied for .local, and the more recently for .onion.

When the [RFC6761] process was put in place, many thought it would only be used a handful of times. However, a large number of applications have since been made to the IETF. The .onion evaluation took almost a year and has started a massive (and often heated) discussion in the IETF.

[RFC6761] has a number of issues. This document groups those issues in several categories.

3. Issues with RFC 6761 process

a. [RFC6761] does not mention if the protocol using the reserved name should be published as an RFC document.

   Most applications have, so far, come from outside organizations, and the described protocols that have not been developed by the IETF.

b. [RFC6761] does not provide clear enough directions as to which group of people is responsible for carrying out the evaluation for inclusion in the registry.

   There are ambiguities and no formal criteria on how the IETF can (or even whether the IETF should) evaluate the merits of applicants to [RFC6761] reservations.

c. The "seven questions" of [RFC6761] are inadequate for evaluating proposals.

   Section 5 of [RFC6761] describes seven questions to be answered by an applicant for [RFC6761] status. However, running this process for the .onion application showed that those seven questions are inadequate for making the determination of whether a particular string qualifies for [RFC6761] treatment.

d. Some organizations may want to experiment with a reserved name.

They may or may not be ready (or willing) to go through a cumbersome process and find [RFC6761] too difficult to deal with. They would like a much simpler registration process, with limited or no burden to apply.

e. The [RFC6761] process could be abused.

In particular, the process can be abused to reserve a specific string within an existing suffix (or set of suffixes using wildcards) not under the control of IETF.

f. [RFC6761] does not have provision for subsequent management of the registry, such as updates, deletions of entries, etc...

4. Issues with the RFC6761 registry

a. The [RFC6761] registry lacks sufficient documentation supporting a registration.

The registry may point to the RFC creating the reservation, but not to the supporting materials.

b. The [RFC6761] registry lacks clear directions for applications to select which name resolution method to use upon seeing the special name.

In particular, it lists the reserved names but does not include direct guidance, neither in free text form nor in machine-readable instructions, for any of the seven audiences. Instead, the registry relies on a reference for each entry to the document that requested its insertion. Such documents could be difficult to read for many readers; for example, [RFC6762] is a 70-page protocol specification which is not an effective way to set expectations of non-technical end-users.

c. Reserving a string in [RFC6761] does not guarantee queries will not leak in the DNS.

The applicants for [RFC6761] status cannot be guaranteed that leakage will not occur and will need to take this into account in their protocol design.

d. The intended usage or protocol for which the [RFC6761] reservation is made may or may not be successful.

In the case of failure of adoption, the reserved string would be wasted.

5. Issues Around the IETF Evaluation of Candidate Strings

a. The IETF does not have a formal process to evaluate candidate strings for issues such as trademark infringement and name collisions.

   This leads to concerns about liability risks incurred by adding a particular string to the [RFC6761] registry.

b. Submitting the application to IETF last call does not guarantee such issues will be always caught.

   [RFC7788] describing the "home networking control protocol" was recently published. That document includes text instructing devices to use names terminating by default with the .home suffix. [RFC7788] did not reference [RFC6761] anywhere and had no IANA sections about this reservation. It was published without anyone noticing this during the entire review process. The issue was caught after the publication, and an errata was published.

6. Other Issues Related to RFC6761

[RFC6761] does not include a mechanism to collaborate with ICANN.

The current round of ICANN gTLD (described at [NEW-GTLD]) is, as the time of this writing, closed to new applications. It is, however, not yet completed. Some applications are still under consideration. There is a risk that, without proper collaboration between the IETF and ICANN, a new entry in the [RFC6761] registry could conflict with one of those applications still under review. It should also be noted that discussions have started about forming the next round of ICANN gTLDs, thus this issue will not go away at the conclusion of the current gTLD round.

7. Security Considerations

This document aims to provide a problem statement that will inform future work. While security and privacy are fundamental considerations, this document expects that future work will include such analysis, and hence no attempt is made to do so here. See [SAC-057] for further considerations.

Reserving names has been presented as a way to prevent leakage into the DNS. However, instructing resolvers to not forward the queries (and/or by instructing authoritative servers not to respond) is not a guarantee that such leakage will be prevented. The security (or

privacy) of an application MUST NOT rely on names not being exposed to the Internet DNS resolution system.

8. IANA Considerations

This document has no IANA actions.

9. Acknowledgements

Thanks to Paul Hoffman for a large amount of editing.

10. References

10.1. Normative References

[IANA-SPECIAL-USE] IANA, "Special-Use Domain Names", 2016, <https://www.iana.org/assignments/special-use-domain-names>.

[RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of Understanding Concerning the Technical Work of the Internet Assigned Numbers Authority", RFC 2860, DOI 10.17487/RFC2860, June 2000, <http://www.rfc-editor.org/info/rfc2860>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <http://www.rfc-editor.org/info/rfc6761>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <http://www.rfc-editor.org/info/rfc7788>.

10.2. Informative References

[DONA] DONA, "DONA Foundation", June 2016, <https://www.dona.net>.

[FROGANS] Frogans Technology, "Frogans Technology", June 2016, <https://www.frogans.org>.

[GUIDEBOOK]
ICANN, "gTLD Application Guidebook", June 2012, <https://newgtlds.icann.org/en/applicants/agb/guidebook-full-04jun12-en.pdf>.

[HUSTON] Huston, G., "What's in a Name?", December 2015, <http://www.circleid.com/posts/20151222_whats_in_a_name/>.

[I-D.lewis-domain-names]
Lewis, E., "Domain Names", draft-lewis-domain-names-03 (work in progress), June 2016.

[NEW-GTLD]
ICANN, "New Generic Top-Level Domains", 2016, <https://newgtlds.icann.org/>.

[SAC-057] ICANN Security and Stability Advisory Committee, "SSAC Advisory on Internal Name Certificates", March 2013, <https://www.icann.org/en/system/files/files/sac-057-en.pdf>.

Appendix A. Editorial Notes

This section (and sub-sections) to be removed prior to publication.

A.1. Venue

An appropriate forum for discussion of this draft is, for now, the DNSOP WG.

A.2. Change History

A.2.1. draft-adpkja-special-names-problem-00

Initial draft circulated for comment.

Appendix B. Change history

[RFC Editor: Please remove this section before publication]

-06 to -05:

- Clarify the sole focus of this draft is to document problems created by RFC6761, and not the larger issue of NON-DNS TLDs.
- Split issue lists and rewrite some for clarity.

-05 to -04:

- Added two issues to the issue list: market failure and experimental use.

-04 to -03:

- Minor edits to correct grammar, clarify that the current ICANN gTLD round is closed.

-03 to -02:

- Significant readability changes to focus the discussion.

-01 to -02:

- A very large number of readability / grammar / reference fixes from Paul Hoffman.

-00 to -01:

- Significant readability changes.

-00:

- Initial draft circulated for comment.

Authors' Addresses

Geoff Huston
APNIC

Email: gih@apnic.net

Peter Koch
DENIC eG
Kaiserstrasse 75-77
Frankfurt 60329
Germany

Email: pk@denic.de

Alain Durand
ICANN

Email: alain.durand@icann.org

Warren Kumari
Google

Email: warren@kumari.net

DNSEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

R. Bellis
ISC

July 02, 2018

DNS Multiple QTYPEs
draft-bellis-dnsext-multi-qtypes-06

Abstract

This document specifies a method for a DNS client to request additional DNS record types to be delivered alongside the primary record type specified in the question section of a DNS query.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 3, 2019.

Table of Contents

1. Introduction

A commonly requested DNS [RFC1035] feature is the ability to receive multiple related resource records (RRs) in a single DNS response.

For example, it may be desirable to receive both the A and AAAA records for a domain name together, rather than having to issue multiple queries.

The DNS wire protocol in theory supports having multiple questions in a single packet, but in practise this does not work:

- Each question consists of the tuple (QNAME, QTYPE, QCLASS). Since each question has its own QNAME field it would be possible for one name to exist and another to not exist, resulting in an inconsistent response code.

- The idea that only a single question is allowed is sufficiently entrenched that many DNS servers will simply return an error (or fail to response at all) if they receive a query with a question count (QDCOUNT) of more than one.

To resolve both of these issues, this document constraints the problem to those cases where only the QTYPE varies by specifying a new option for the Extension Mechanisms for DNS (EDNS [RFC6891]) that contains an additional list of QTYPE values that the client wishes to receive in addition to that in the primary question.

TODO: why not "ANY" ?

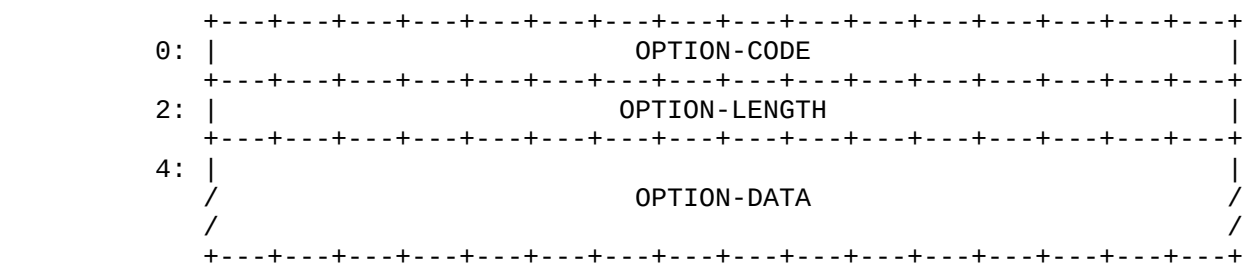
2. Terminology used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Description

3.1. Multiple QTYPE EDNS Option Format

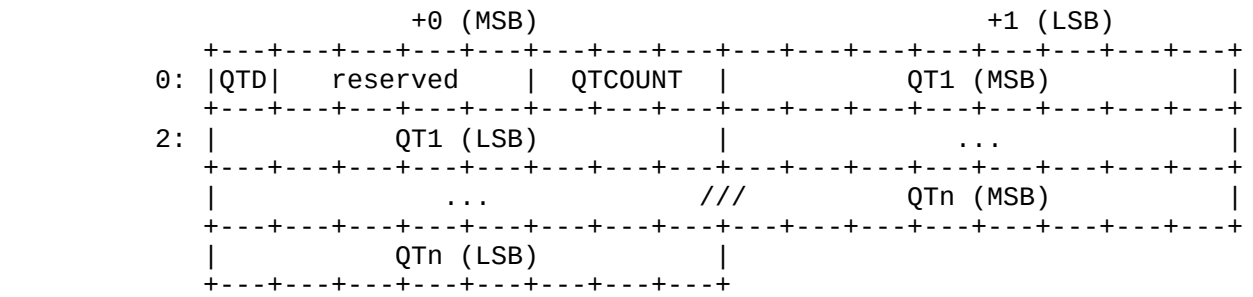
The overall format of an EDNS option is shown for reference below, per [RFC6891], followed by the option specific data:



OPTION-CODE: TBD by IANA

OPTION-LENGTH: Size (in octets) of OPTION-DATA.

OPTION-DATA: Option specific, as below:



QTD: this bit indicates the direction of the packet. It MUST be clear (0) in a query and set (1) in a response.

QTCOUNT: a 3 bit field with range 0 .. 7 specifying the number of QT fields to follow. NB: Whilst the QTCOUNT could in theory be

calculated based on the OPTION-LENGTH field, having it explicitly specified ensures a sensible constraint its value.

QTn: a 2 byte field (MSB first) specifying a DNS RR type. The RR type MUST be for a real resource record, and MUST NOT refer to a pseudo RR type such as "OPT", "IXFR", "TSIG", "*", etc.

3.2. Response Generation

3.2.1. Server Side Processing

A conforming server that receives a Multiple QTYPE Option in a query MUST return a Multiple QTYPE Option in its response.

The QTD bit in that response MUST be set (1) as protection against servers which simply echo unknown EDNS options verbatim. If the QTD bit in a response is zero the client MUST treat the response as if this option is unsupported.

The server SHOULD attempt to return any resource records known to it that match the additional (QNAME, QTn, QCLASS) tuples. These records MUST be returned in the Answer Section of the response, but the answer for the primary QTYPE from the Question Section MUST be included first.

For any particular QTn in the query, if the server provides additional answers, or has knowledge that the RR type type does not exist for that QNAME (a "negative answer"), it must include that QTn value in the Multiple QTYPE Option of its response.

A negative answer is therefore indicated by the combination of the presence of a QTn value in the Multiple QTYPE Option and the absence of a matching record in the Answer Section. This is necessary (in the absence of DNSSEC) to differentiate between absence of the record from the zone and absence of the record from the response.

A server that is authoritative for the specified QNAME on receipt of a Multiple QTYPE Option MUST attempt to return all specified RR types except where that would result in truncation in which case it may omit some (or all) of the records for the additional RR types. Those RR types MUST then also be omitted from the Multiple QTYPE Option in the response.

A caching recursive server receiving a Multiple QTYPE Option SHOULD attempt to fill its positive and negative caches with all of the specified RR types before returning its response to the client.

TODO: is there a case for mandatory answers, i.e. the client saying I _really_ want all these?

3.2.2. Client Side Processing

Recursive resolvers MAY use this method to obtain multiple records from an authoritative server. For the purposes of Section 5.4.1 of [RFC2181] any authoritative answers received MUST be ranked the same as the answer for the primary question.

3.2.3. DNSSEC

If the DNS client sets the "DNSSEC OK" (DO) bit in the query then the server MUST also return the related DNSSEC records that would have been returned in a standalone query for the same QTYPE.

A negative answer from a signed zone MUST contain the appropriate authenticated denial of existence records, per [RFC4034] and [RFC5155].

In a signed zone there is a theoretical risk of valid signatures for one RR type and invalid signatures for another. This is the only case known to the author where the response code for any particular QNAME may be inconsistent across different RR types.

Should a validating resolver produce NOERROR for some RR types and SERVFAIL for others it MUST omit the RR types that failed to validate from its response and from the QTn fields on the Multiple QTYPE option. The client MAY then initiate standalone queries for those RR types.

4. Security Considerations

The method documented here does not change any of the security properties of the DNS protocol itself.

It should however be noted that this method does increase the potential amplification factor when the DNS protocol is used as a vector for a denial of service attack.

5. IANA Considerations

IANA is requested to assign a new value in the DNS EDNS0 Option Codes registry.

6. Acknowledgements

The author wishes to thank the following for their feedback and reviews during the initial development of this document: Michael Graff, Olafur Gudmundsson, Matthijs Mekking, Paul Vixie.

7. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Author’s Address

Ray Bellis
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City CA 94063
USA

Phone: +1 650 423 1200
Email: ray@isc.org

DNSOP Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 29, 2017

R. Bellis
ISC
S. Cheshire
Apple Inc.
J. Dickinson
S. Dickinson
Sinodun
A. Mankin
Salesforce
T. Pusateri
Unaffiliated

July 28, 2016

DNS Session Signaling
draft-bellis-dnsop-session-signal-02

Abstract

The EDNS(0) Extension Mechanism for DNS [RFC6891] is explicitly defined to only have "per-message" semantics. This document defines a new Session Signaling Opcode used to carry persistent "per-session" type-length-values (TLVs), and defines an initial set of TLVs used to manage session timeouts and termination.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 29, 2017.

Copyright Notice

Table of Contents

1. Introduction

The use of transports other than UDP for DNS is being increasingly specified, for example, DNS-over-TCP [RFC7766], DNS-over-TLS [RFC7858] and recent work on DNS Push Notifications [I-D.ietf-dnssd-push]. Such transports frequently use persistent, long-lived sessions and therefore when using them for transporting DNS messages it is of benefit to have a mechanism that can establish parameters associated with those sessions, such as timeouts. In such situations it is also advantageous to support server initiated messages.

The EDNS(0) Extension Mechanism for DNS [RFC6891] is explicitly defined to only have "per-message" semantics. Whilst EDNS(0) has been used to signal at least one session related parameter (the EDNS TCP KeepAlive option [RFC7828]) the result is less than optimal due to the restrictions imposed by the EDNS(0) semantics and the lack of server initiated signalling. This document defines a new Session Signaling Opcode used to carry persistent "per-session" type-length-values (TLVs), and defines an initial set of TLVs used to manage session timeouts and termination.

With EDNS(0), multiple options may be packed into a single OPT RR, and there is no generalized mechanism for a client to be able to tell whether a server has processed or otherwise acted upon each individual option within the combined OPT RR. The specifications for each individual option need to define how each different option is to be acknowledged, if necessary.

With Session Signaling, in contrast, each Session Signaling operation is communicated in its own separate DNS message, and the RCODE in the response indicates the success or failure of that operation.

It should be noted that the message format for Session Signaling operations (see Section 3.2) differs from the DNS packet format used for standard queries and responses, in that it has a shorter header (four octets instead of usual twelve octets).

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC2119].

The terms "initiator" and "responder" correspond respectively to the initial sender and subsequent receiver of a Session Signaling TLV, regardless of which was the "client" and "server" in the usual DNS sense.

The term "sender" may apply to either an initiator or responder.

The term "session" in the context of this document means the exchange of DNS messages using an end-to-end transport protocol where:

- The connection between client and server is persistent and relatively long-lived (i.e. minutes or hours, rather than seconds).

- Either end of the connection may initiate messages to the other

- Messages are delivered in order

3. Protocol Details

Session Signaling messages MUST only be carried in protocols and in environments where a session may be established according to the definition above. Standard DNS over TCP [RFC1035], and DNS over TLS [RFC7858] are suitable protocols. DNS over plain UDP is not appropriate since it fails on the requirement for in-order message delivery, and, in the presence of NAT gateways and firewalls with short UDP timeouts, it fails to provide a persistent bi-directional communication channel unless an excessive amount of keepalive traffic is used.

Session Signaling messages relate only to the specific session in which they are being carried. Where a middle box (e.g. a DNS proxy, forwarder, or session multiplexer) is in the path the message MUST NOT be blindly forwarded in either direction by that middle box. This does not preclude the use of these messages in the presence of a NAT box that rewrites IP-layer or transport-layer headers but otherwise maintains the effect of a single session.

A client MAY attempt to initiate Session Signaling messages at any time on a connection; receiving a NOTIMP response in reply indicates that the server does not implement Session Signaling, and the client SHOULD NOT issue further Session Signaling messages on that connection.

A server SHOULD NOT initiate Session Signaling messages until a client-initiated Session Signaling message is received first, unless in an environment where it is known in advance by other means that both client and server support Session Signaling. This requirement is to ensure that the clients that do not support Session Signaling do not receive unsolicited inbound Session Signaling messages that they would not know how to handle.

3.1. Session Lifecycle

A session begins when a client makes a new connection to a server.

The client may perform as many DNS operations as it wishes on the newly created connection. Operations SHOULD be pipelined (i.e., the client doesn’t need wait for a reply before sending the next message). The server MUST act on messages in the order they are received, but responses to those messages MAY be sent out of order, if appropriate.

When a server implementing this specification receives a new connection from a client, it MUST begin by internally assigning an initial idle timeout of 30 seconds to that connection. At both servers and clients, the generation or reception of any complete DNS message, including DNS requests, responses, updates, or Session Signaling messages, resets the idle timer for that connection (see [RFC7766]).

If, at any time during the life of the connection, half the idle-timeout value (i.e., 15 seconds by default) elapses without any DNS messages being sent or received on a connection, then the connection is considered stale and the client MUST take action. When this happens the client MUST either send at least one new message to reset the idle timer - such as a Session Signaling Idle Timeout message (see Section 4.2.1), or any other valid DNS message - or close the connection.

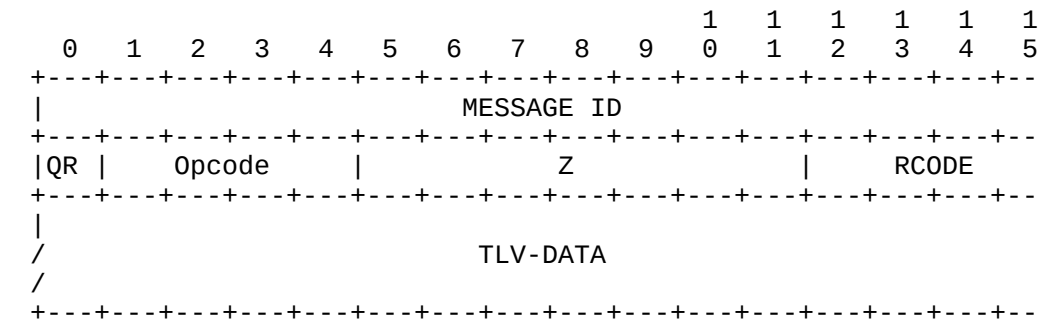
If, at any time during the life of the connection, the full idle-timeout value (i.e., 30 seconds by default) elapses without any DNS messages being sent or received on a connection, then the connection is considered delinquent and the server SHOULD forcibly terminate the connection. For sessions over TCP (or over TLS-over-TCP), to avoid the burden of having a connection in TIME-WAIT state, instead of closing the connection gracefully with a TCP FIN the server SHOULD abort the connection with a TCP RST. (In the Sockets API this is achieved by setting the SO_LINGER option to zero before closing the socket.)

If the client wishes to keep an idle connection open for longer than the default duration without having to send traffic every 15 seconds, then it uses the Session Signaling Idle Timeout message to request a longer idle timeout (see Section 4.2.1).

A client is not required to wait until half of the idle-timeout value before closing a connection. A client MAY close a connection at any time, at the client’s discretion, if it has no further need for the connection at that time.

3.2. Message Format

A Session Signaling message begins with the first 4 octets of the standard DNS message header [RFC1035], with the Opcode field set to the Session Signaling Opcode. A Session Signaling message does not contain the QDCOUNT, ANCOUNT, NSCOUNT and ARCOUNT fields fields used in standard DNS queries and responses. This 4-octet header is followed by a single Session Signaling TLV.



The MESSAGE ID, QR, Opcode and RCODE fields have their usual meanings [RFC1035].

The Z bits are currently unused, and SHOULD be set to zero (0) in requests and responses unless re-defined in a later specification.

3.3. Message Handling

On a connection between a client and server that support Session Signaling, either end may unilaterally send Session Signaling messages at any point in the lifetime of a session, and therefore either client or server may be the initiator of a message. The initiator MUST set the value of the QR bit in the DNS header to zero (0), and the responder MUST set it to one (1).

<<TODO: Specify behaviour on receipt of a message from an initiator with the QR bit set to 1, etc.>>

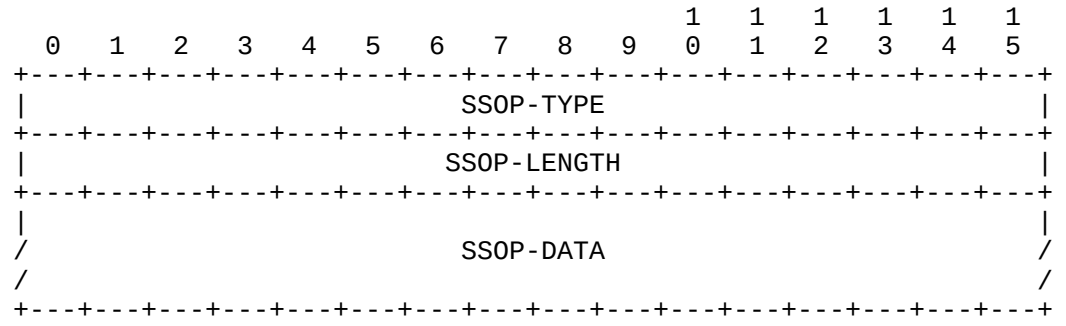
Every Session Signaling request message MUST elicit a response (which MUST have the same ID in the DNS message header as in the request).

<< RB: should the presence of a SS message create a "sequencing point", such that all pending responses must be answered? SC: I do not believe so. We can define an explicit SS "sequencing point" opcode for this if necessary. >>

The RCODE value in a response uses a subset of the standard (non-extended) RCODE values from the IANA DNS RCODEs registry, interpreted as follows:

Code	Mnemonic	Description
0	NOERROR	TLV processed successfully
1	FORMERR	TLV format error
4	NOTIMP	Session Signaling not supported
5	REFUSED	TLV declined for policy reasons

3.4. TLV Format



SSOP-TYPE: A 16 bit field in network order giving the type of the current Session Signaling TLV per the IANA DNS Session Signaling Type Codes Registry.

<< SC: I changed SESSION-TYPE to SSOP-TYPE because to me SESSION-TYPE read as "type of session" which it is not. It is Session Signaling Operation Type, Session Signaling Operation Length, Session Signaling Operation Data. >>

SSOP-LENGTH: A 16 bit field in network order giving the size in octets of SSOP-DATA.

SSOP-DATA: Type-code specific.

4. Mandatory TLVs

4.1. Session Management Support TLVs

4.1.1. "Not Implemented"

Since the "NOTIMP" RCODE in the DNS message header is used to indicate lack of support for the Session Signaling Opcode itself, a different mechanism is used to indicate lack of support of a specific SSOP-TYPE. If a server that supports Session Signaling receives a Session Signaling query message (QR bit zero) with an SSOP-TYPE it does not support, it returns a response message (QR bit one) containing a single Session Signaling SSOP-NOTIMP TLV (0). The MESSAGE ID in the message header serves to tell the client which of its requests was not understood.

The SSOP-NOTIMP TLV has no SSOP-DATA.

4.2. Session Management TLVs

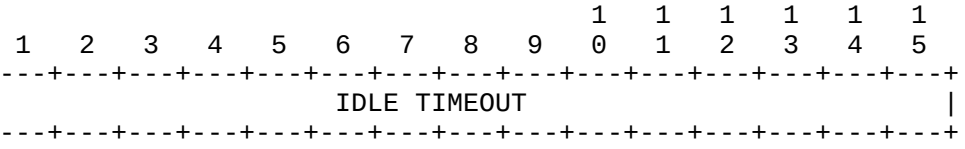
4.2.1. Idle Timeout

The Idle Timeout TLV (1) is be used by a client to reset a connection’s idle timer, and at the same time to request what the idle timeout should be from this point forward in the connection.

The Idle Timeout TLV also MAY be initiated by a server, to unilaterally inform the client of a new idle timeout this point forward in this connection.

It is not required that the Idle Timeout TLV be used in every session. While many Session Signaling operations (such as DNS Push Notifications [I-D.ietf-dnssd-push]) will be used in conjunction with a long-lived connection, this is not required, and in some cases the default 30-second timeout may be perfectly appropriate.

The SSOP-DATA for the the Idle Timeout TLV is as follows:



IDLE TIMEOUT: the idle timeout for the current session, specified as a 16 bit word in network order in units of 100 milliseconds. This is the timeout at which the server will forcibly terminate the connection with a TCP RST (or equivalent for other protocols); after half this interval the client MUST take action to either preserve the connection, or close it if it is no longer needed.

In a client-initiated Session Signaling Idle Timeout message, the IDLE TIMEOUT contains the client's requested value for the idle timeout.

In a server response to a client-initiated message, the IDLE TIMEOUT contains the server's chosen value for the idle timeout, which the client MUST respect. This is modeled after the DHCP protocol, where the client requests a certain lease lifetime, but the server is the ultimate authority for deciding what lease lifetime is actually granted.

In a server-initiated Session Signaling Idle Timeout message, the IDLE TIMEOUT unilaterally informs the client of the new idle timeout this point forward in this connection.

In a client response to a server-initiated message, there is no SSOP-DATA. SSOP-LENGTH is zero.

<<TODO: Specify the behaviour when a server sends a 0 IDLE TIMEOUT.>>

The Idle Timeout TLV (3) has similar intent to the EDNS TCP Keepalive Option [RFC7828]. A client/server pair that supports Session Signaling MUST NOT use the EDNS TCP KeepAlive option within any message once bi-directional Session Signaling support has been confirmed.

<< SC: And if client or server does use EDNS TCP Keepalive, then other end should... close connection? Silently ignore? >>

4.2.2. Terminate Session

There may be rare cases where a server is overloaded and wishes to shed load. If the server simply closes connections, the likely behaviour of clients is to detect this as a network failure, and reconnect.

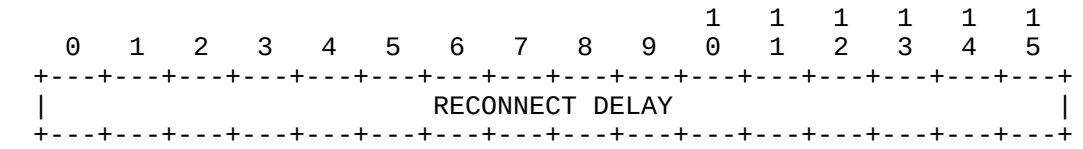
To avoid this reconnection implosion, the server sends a Terminate Session TLV (2) to inform the client of the overload situation. After sending a Terminate Session TLV the server MUST NOT send any further messages on the connection.

The Terminate Session TLV (2) MUST NOT be initiated by a client.

<<TODO: Specify behaviour if the Terminate Session TLV is initiated by a client.>>

Upon receipt of the Terminate Session TLV (2) the client MUST make note of the reconnect delay for this server, and then immediately close the connection.

The SSOP-DATA is as follows:



RECONNECT DELAY: a time value, specified as a 16 bit word in network order in units of 100 milliseconds, within which the client MUST NOT establish a new session to the current server.

The RECOMMENDED value is 10 seconds. << RB: text required here about default values for load balancers, etc >>

5. IANA Considerations

5.1. DNS Session Signaling Opcode Registration

IANA are directed to assign the value TBD for the Session Signaling Opcode in the DNS OpCodes Registry.

5.2. DNS Session Signaling Type Codes Registry

IANA are directed to create the DNS Session Signaling Type Codes Registry, with initial values as follows:

Type	Name	Status	Reference
0	SSOP-NOTIMP	Standard	RFC-TBD1
1	SSOP-Keepalive	Standard	RFC-TBD1
2	SSOP-Terminate	Standard	RFC-TBD1
3 - 63	Unassigned, reserved for session management TLVs		
64 - 63487	Unassigned		
63488 - 64511	Reserved for local / experimental use		
64512 - 65535	Reserved for future expansion		

Registration of additional Session Signaling Type Codes requires Expert Review. << RB: definition of process required? >>

6. Security Considerations

If this mechanism is to be used with DNS over TLS, then these messages are subject to the same constraints as any other DNS over TLS messages and MUST NOT be sent in the clear before the TLS session is established.

7. Acknowledgements

TBW

8. References

8.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

[RFC7766] Dickinson, J., Dickinson, S., Bellis, R., Mankin, A., and D. Wessels, "DNS Transport over TCP - Implementation Requirements", RFC 7766, DOI 10.17487/RFC7766, March 2016, <http://www.rfc-editor.org/info/rfc7766>.

[RFC7828] Wouters, P., Abley, J., Dickinson, S., and R. Bellis, "The edns-tcp-keepalive EDNS0 Option", RFC 7828, DOI 10.17487/RFC7828, April 2016, <http://www.rfc-editor.org/info/rfc7828>.

8.2. Informative References

[I-D.ietf-dnssd-push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-08 (work in progress), July 2016.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <http://www.rfc-editor.org/info/rfc7858>.

Authors' Addresses

Ray Bellis
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City CA 94063
USA

Phone: +1 650 423 1200
Email: ray@isc.org

Stuart Cheshire
Apple Inc.
1 Infinite Loop
Cupertino CA 95014
USA

Phone: +1 408 974 3207
Email: cheshire@apple.com

John Dickinson
Sinodun Internet Technologies
Magadalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: jad@sinodun.com

Sara Dickinson
Sinodun Internet Technologies
Magadalen Centre
Oxford Science Park
Oxford OX4 4GA
United Kingdom

Email: sara@sinodun.com

Allison Mankin
Salesforce

Email: allison.mankin@gmail.com

Tom Pusateri
Unaffiliated

Phone: +1 843 473 7394
Email: pusateri@bangj.com

Network Working Group
Internet-Draft
Updates: 7050 (if approved)
Intended status: Standards Track
Expires: September 20, 2020

S. Cheshire
Apple Inc.
D. Schinazi
Google LLC

March 19, 2020

Special Use Domain Name 'ipv4only.arpa' draft-cheshire-sudn-ipv4only-dot-arpa-17

Abstract

The specification for how a client discovers its local network's NAT64 prefix (RFC7050) defines the special name 'ipv4only.arpa' for this purpose, but in its Domain Name Reservation Considerations section that specification indicates that the name actually has no particularly special properties that would require special handling, and does not request IANA to record the name in the Special-Use Domain Names registry.

Consequently, despite the well articulated special purpose of the name, 'ipv4only.arpa' was not recorded in the Special-Use Domain Names registry as a name with special properties.

This document describes the special treatment required, formally declares the special properties of the name, adds similar declarations for the corresponding reverse mapping names, and updates RFC7050.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 20, 2020.

Table of Contents

1. Introduction

The specification for how a client discovers its local network's NAT64 prefix [RFC7050] defines the special name 'ipv4only.arpa' for this purpose, but in its Domain Name Reservation Considerations section that specification indicates that the name actually has no particularly special properties that would require special handling, and does not request IANA to record the name in the Special-Use Domain Names registry [SUDN].

Consequently, despite the well articulated special purpose of the name, 'ipv4only.arpa' was not recorded in the Special-Use Domain Names registry [SUDN] as a name with special properties.

This omission was discussed in the Special-Use Domain Names Problem Statement [RFC8244].

As a result of this omission, in cases where software needs to give this name special treatment in order for it to work correctly, there was no clear mandate authorizing software authors to implement that special treatment. Software implementers were left with the choice between not implementing the special behavior necessary for the name queries to work correctly, or implementing the special behavior and being accused of being noncompliant with some RFC.

This document describes the special treatment required, formally declares the special properties of the name, and adds similar declarations for the corresponding reverse mapping names.

1.1. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Specialness of 'ipv4only.arpa'

The hostname 'ipv4only.arpa' is peculiar in that it was never intended to be treated like a normal hostname.

A typical client never has any reason to look up the IPv4 address records for 'ipv4only.arpa'. No normal user is ever trying to view a web site hosted at that domain name, or trying to send email to an email address at that domain name. The name 'ipv4only.arpa' is already known, by IETF specification [RFC7050], to have exactly two

IPv4 address records, 192.0.0.170 and 192.0.0.171. No client ever has to look up the name in order to learn those two addresses.

In contrast, clients often look up the IPv6 AAAA address records for 'ipv4only.arpa', which is contrary to general DNS expectations, given that it is already known, by IETF specification [RFC7050], that 'ipv4only.arpa' is an IPv4-only name, which has no IPv6 AAAA address records. And yet, clients expect to receive, and do in fact receive, positive answers for these IPv6 AAAA address records that apparently should not exist.

This odd query behaviour comes not because clients are using DNS to learn legitimate answers from the name's legitimate authoritative server. Instead, the DNS protocol has, in effect, been co-opted as an improvised client-to-middlebox communication protocol, to look for a DNS64/NAT64 [RFC6146] [RFC6147] gateway and, if one is present, to request that it disclose the prefix it is using for IPv6 address synthesis.

This use of specially crafted DNS queries as an improvised client-to-middlebox communication protocol has a number of specific consequences, outlined below, which client software needs to take into account if the queries are to produce the desired results, particularly when used on a multi-homed host, or when a VPN tunnel is in use. The name 'ipv4only.arpa' is most definitely a special name, and needs to be listed in IANA's registry along with other DNS names that have special uses [SUDN].

3. Consequences of 'ipv4only.arpa' not being declared special

As a result of the original specification [RFC7050] not formally declaring 'ipv4only.arpa' to have special properties, there was no clear mandate for DNS software to treat this name specially. In particular, this lack of mandate for special treatment is relevant (a) to the name resolution APIs and libraries on client devices, and (b) to DNS64 [RFC6147] implementations. These two aspects are discussed in more detail below.

3.1. Consequences for Name Resolution APIs and Libraries

A serious problem can occur with DNS64/NAT64 when a device is configured to use a recursive resolver other than the one it learned from the network.

A device joining a NAT64 network will learn the recursive resolver recommended for that network, typically via IPv6 Router Advertisement Options for DNS Configuration [RFC8106] or via DNS Configuration options for DHCPv6 [RFC3646]. On a NAT64 network it is essential

that the client use the DNS64 recursive resolver recommended for that network, since only that recursive resolver can be relied upon to know the appropriate prefix(es) to use for synthesizing IPv6 addresses that will be acceptable to that NAT64 gateway.

However, it is becoming increasingly common for users to manually override their default DNS configuration because they wish to use some other public recursive resolver on the Internet, which may offer better speed, better reliability, or better privacy than the local network's default recursive resolver. At the time of writing, examples of widely known public recursive resolver services include Cloudflare Public DNS [DNS1], Google Public DNS [DNS8], and Quad9 [DNS9].

Another common scenario is the use of corporate or personal VPN client software. Both for privacy reasons, and also because the local network's recursive resolver will typically be unable to provide answers for the company's private internal host names, so VPN client software overrides the local network's default configuration, to divert some or all DNS requests to the company's own private internal recursive resolver, reached through the VPN tunnel. As with the case described above of public recursive resolver services, the company's private internal recursive resolver cannot be expected to be able to synthesize IPv6 addresses correctly for use with the local network's NAT64 gateway, because the company's private internal recursive resolver is unlikely to be aware of the NAT64 prefix in use on the NAT64 network to which the client device is currently attached. It is clear that a single recursive resolver cannot meet both needs. The local network's recursive resolver cannot give answers for some company's private internal host names, and some company's private internal recursive resolver cannot give correctly synthesized IPv6 addresses suitable for the local network's NAT64 gateway.

Note that multiple NAT64 services may be simultaneously available to a client. For example, the local network may provide NAT64 service (to allow a IPv6-only client device to access IPv4-only Internet services), while at the same time a corporate VPN may also provide NAT64 service (to allow a client connecting via an IPv6-only VPN tunnel to access IPv4-only corporate services). The NAT64 address synthesis prefixes for the two NAT64 services may be different. In this case it is essential that the NAT64 address synthesis prefix used on the local network be the prefix learned from the local network's recursive resolver, and the NAT64 address synthesis prefix used on the VPN tunnel be the prefix learned from the VPN tunnel's recursive resolver.

The conflict here arises because DNS is being used for two unrelated purposes. The first purpose is retrieving data from a (nominally) global database -- generally retrieving the IP address(es) associated with a hostname. The second purpose is using the DNS protocol as a middlebox communication protocol, to interrogate the local network infrastructure to discover the IPv6 prefix(es) in use by the local NAT64 gateway for address synthesis.

3.2. Consequences for DNS64 Implementations

As a result of there being no mandate for special treatment, queries for 'ipv4only.arpa' had to be handled normally, resulting in DNS64 gateways performing unnecessary IPv6 address record queries (DNS qtype "AAAA", always returning negative responses) and IPv4 address record queries (DNS qtype "A", always returning the same positive responses) to the authoritative 'arpa' name servers.

Having DNS64 gateways around the world issue these queries generated additional load on the authoritative 'arpa' name servers, which was redundant when the name 'ipv4only.arpa' is defined, by IETF specification [RFC7050], to have exactly two IPv4 address records, 192.0.0.170 and 192.0.0.171, and no other IPv4 or IPv6 address records.

Also, at times, for reasons that remain unclear, the authoritative 'arpa' name servers have been observed to be slow or unresponsive. The failures of these 'ipv4only.arpa' queries result in unnecessary failures of the DNS64 gateways and of the client devices that depend on them for DNS64 [RFC6147] address synthesis.

Even when the authoritative 'arpa' name servers are operating correctly, having to perform an unnecessary query to obtain an answer that is already known in advance can add precious milliseconds of delay, affecting user experience on the client devices waiting for those synthesized replies.

4. Remedies

This document leverages operational experience to update the Domain Name Reservation Considerations [RFC6761] section of the earlier specification [RFC7050] with one that more accurately lists the actual special properties of the name 'ipv4only.arpa', so that software can legitimately implement the correct behavior necessary for better performance, better reliability, and correct operation.

These changes affect two bodies of software, (a) the name resolution APIs and libraries on client devices, and (b) DNS64 implementations.

The new special rules specified in this document for name resolution APIs and libraries state how they should select which recursive resolver to query to learn the IPv6 address synthesis prefix in use on a particular physical or virtual interface. Specifically: When querying for the name 'ipv4only.arpa', name resolution APIs and libraries should use the recursive resolver recommended by the network for the interface in question, rather than a recursive resolver configured manually, a recursive resolver configured by VPN software, or a full-service recursive resolver running on the local host. Superficially this might seem like a security issue, since the user might have explicitly configured the particular DNS resolver they wish to use, and rather than using that, the name resolution code ignores the user's stated preference and uses untrusted input received from the network instead. However, the 'ipv4only.arpa' query is not really a DNS query in the usual sense; even though it may look like a DNS query, it is actually an improvised client-to-middlebox communication protocol in disguise. For NAT64 to work at all, it is necessary for the interface on which NAT64 translation is being performed to tell the host the address of the DNS64 recursive resolver the host must use to learn the NAT64 prefix being used by that NAT64. This is typically done via IPv6 Router Advertisement Options for DNS Configuration [RFC8106] or via DNS Configuration options for DHCPv6 [RFC3646].

The new special rules specified in this document for DNS64 implementations recommend that they avoid performing run-time network queries for values that are known to be fixed by specification.

A useful property of the way NAT64 Prefix Discovery [RFC7050] was originally specified was that it allowed for incremental deployment. Even if existing DNS64 gateways, that were unaware of the special 'ipv4only.arpa' name, were already deployed, once IANA created the appropriate 'ipv4only.arpa' records, clients could begin to use the new facility immediately. Clients could send their special queries for 'ipv4only.arpa' to an ipv4only-unaware DNS64 gateway, and (after a query to IANA's servers) the DNS64 gateway would then generate the correct synthesized response.

While this was a useful transition strategy to enable rapid adoption, it is not the ideal end situation. For better performance, better reliability, and lower load in IANA's servers, it is preferable for DNS64 gateways to be aware of the special 'ipv4only.arpa' name so that they can avoid issuing unnecessary queries. Network operators who wish to provide reliable, high performance service to their customers are motivated to prefer DNS64 gateways that recognize the special 'ipv4only.arpa' name and apply the appropriate optimizations.

5. Security Considerations

One of the known concerns with DNS64 is that it conflicts with DNSSEC. If DNSSEC is used to assert cryptographically that a name has no IPv6 AAAA records, then this interferes with using DNS64 address synthesis to assert that those nonexistent IPv6 AAAA records do exist.

Section 3 of the DNS64 specification [RFC6147] discusses this:

> ... DNS64 receives a query with the DO bit set and the CD bit set. In this case, the DNS64 is supposed to pass on all the data it gets to the query initiator. This case will not work with DNS64, unless the validating resolver is prepared to do DNS64 itself.

The NAT64 Prefix Discovery specification [RFC7050] provides the mechanism for the query initiator to learn the NAT64 prefix so that it can do its own validation and DNS64 synthesis as described above. With this mechanism the client can (i) interrogate the local DNS64/NAT64 gateway with an 'ipv4only.arpa' query to learn the IPv6 address synthesis prefix, (ii) query for the (signed) IPv4 address records itself, and validate the response, and then (iii) perform its own IPv6 address synthesis locally, combining the IPv6 address synthesis prefix learned from the local DNS64/NAT64 gateway with the validated DNSSEC-signed data learned from the global Domain Name System.

It is conceivable that over time, if DNSSEC adoption continues to grow, the majority of clients could move to this validate-and-synthesize-locally model, which reduces the DNS64 machinery to the vestigial role of simply responding to the 'ipv4only.arpa' query to report the local IPv6 address synthesis prefix. In no case does the client care what answer(s) the authoritative 'arpa' name servers might give for that query. The 'ipv4only.arpa' query is being used purely as a local client-to-middlebox communication message.

This approach is even more attractive if it does not create an additional dependency on the authoritative 'arpa' name servers to answer a query that is unnecessary because the DNS64/NAT64 gateway already knows the answer before it even issues the query. Avoiding this unnecessary query improves performance and reliability for the client, and reduces unnecessary load for the authoritative 'arpa' name servers.

Hard-coding the known answers for 'ipv4only.arpa' IPv4 address record queries (DNS qtype "A") in recursive resolvers also reduces the risk of malicious devices intercepting those queries and returning incorrect answers. Because the 'ipv4only.arpa' zone has to be an

insecure delegation (see below) DNSSEC cannot be used to protect these answers from tampering by malicious devices on the path.

With respect to the question of whether 'ipv4only.arpa' should be a secure or insecure delegation, we need to consider two paths of information flow through the network: The path from the server authoritative for 'ipv4only.arpa' to the DNS64 recursive resolver, and the path from the DNS64 recursive resolver to the ultimate client.

The path from the authoritative server to the DNS64 recursive resolver (queries for IPv4 address records) need not be protected by DNSSEC, because the DNS64 recursive resolver already knows, by specification, what the answers are. In principle, if this were a secure delegation, and 'ipv4only.arpa' were a signed zone, then the path from the authoritative server to the DNS64 recursive resolver would still work, but DNSSEC is not necessary here. Run-time cryptographic signatures are not needed to verify compile-time constants. Validating the signatures could only serve to introduce potential failures into the system for minimal benefit.

The path from the DNS64 recursive resolver to the ultimate client (queries for IPv6 address records) *cannot* be protected by DNSSEC, because the DNS64 recursive resolver is synthesizing IPv6 address answers, and does not possess the DNSSEC secret key required to sign those answers.

Consequently, the 'ipv4only.arpa' zone MUST be an insecure delegation, to give DNS64/NAT64 gateways the freedom to synthesize answers to those queries at will, without the answers being rejected by DNSSEC-capable resolvers. DNSSEC-capable resolvers that follow this specification MUST NOT attempt to validate answers received in response to queries for the IPv6 AAAA address records for 'ipv4only.arpa'. Note that the name 'ipv4only.arpa' has no use outside of being used for this special DNS pseudo-query used to learn the DNS64/NAT64 address synthesis prefix, so the lack of DNSSEC security for that name is not a problem.

The original NAT64 Prefix Discovery specification [RFC7050] stated, incorrectly:

> A signed "ipv4only.arpa." allows validating DNS64 servers (see [RFC6147] Section 3, Case 5, for example) to detect malicious AAAA resource records. Therefore, the zone serving the well-known name has to be protected with DNSSEC.

This document updates the previous specification [RFC7050] to correct that error. The 'ipv4only.arpa' zone MUST be an insecure delegation.

6. IANA Considerations

[Once published] IANA has created an insecure delegation for 'ipv4only.arpa' to allow DNS64 recursive resolvers to create synthesized AAAA answers within that zone.

IANA has recorded the following names in the Special-Use Domain Names registry [SUDN]:

ipv4only.arpa.
170.0.0.192.in-addr.arpa.
171.0.0.192.in-addr.arpa.

IANA has recorded the following IPv4 addresses in the IPv4 Special-Purpose Address Registry [SUv4]:

192.0.0.170
192.0.0.171

7. Domain Name Reservation Considerations

7.1. Special Use Domain Name 'ipv4only.arpa'

The name 'ipv4only.arpa' is defined, by IETF specification [RFC7050], to have two IPv4 address records with rdata 192.0.0.170 and 192.0.0.171.

When queried via a DNS64 [RFC6147] recursive resolver, the name 'ipv4only.arpa' is also defined to have IPv6 AAAA records, with rdata synthesized from a combination of the NAT64 IPv6 prefix(es) and the IPv4 addresses 192.0.0.170 and 192.0.0.171. This can return more than one pair of IPv6 addresses if there are multiple NAT64 prefixes.

The name 'ipv4only.arpa' has no other IPv4 or IPv6 address records. There are no subdomains of 'ipv4only.arpa'. All names falling below 'ipv4only.arpa' are defined to be nonexistent (NXDOMAIN).

The name 'ipv4only.arpa' is special to
(a) client software wishing to perform DNS64 address synthesis,
(b) APIs responsible for retrieving the correct information, and
(c) the DNS64 recursive resolver responding to such requests.
These three considerations are listed in items 2, 3 and 4 below:

1. Normal users should never have reason to encounter the 'ipv4only.arpa' domain name. If they do, they should expect queries for 'ipv4only.arpa' to result in the answers required by the specification [RFC7050]. Normal users have no need to know that 'ipv4only.arpa' is special.

2. Application software may explicitly use the name 'ipv4only.arpa' for DNS64/NAT64 address synthesis, and expect to get the answers required by the specification [RFC7050]. If application software encounters the name 'ipv4only.arpa' in the normal course of handling user input, the application software should resolve that name as usual and need not treat it in any special way.

3. Name resolution APIs and libraries MUST recognize 'ipv4only.arpa' as special and MUST give it special treatment.

   Learning a network's NAT64 prefix is by its nature an interface-specific operation, and the special DNS query used to learn this interface-specific NAT64 prefix MUST be sent to the DNS recursive resolver address(es) the client learned via the configuration machinery for that specific client interface. The NAT64 prefix is a per-interface property, not a per-device property.

   Regardless of any manual client DNS configuration, DNS overrides configured by VPN client software, or any other mechanisms that influence the choice of the client's recursive resolver address(es) (including client devices that run their own local recursive resolver and use the loopback address as their configured recursive resolver address) all queries for 'ipv4only.arpa' and any subdomains of that name MUST be sent to the recursive resolver learned from the network interface in question via IPv6 Router Advertisement Options for DNS Configuration [RFC8106], DNS Configuration options for DHCPv6 [RFC3646], or other configuration mechanisms. Because DNS queries for 'ipv4only.arpa' are actually a special middlebox communication protocol, it is essential that they go to the correct middlebox for the interface in question, and failure to honor this requirement would cause failure of the NAT64 Prefix Discovery mechanism [RFC7050].

   One implication of this is that, on multi-homed devices (devices that allow more than one logical or physical IP interface to be active at the same time, e.g., cellular data and Wi-Fi, or one physical interface plus a VPN connection), clients MUST use interface-aware name resolution APIs. On different (logical or physical) interfaces, different DNS64 answers may be received, and DNS64 answers are only valid for the interface on which they were received. On multi-homed devices (including devices that support VPN), name resolution APIs that do not include interface parameters will not work reliably with NAT64. On single-homed devices, interface-unaware name resolution APIs are acceptable since when only one interface can be active at a time there is no need to specify an interface.

DNSSEC-capable resolvers MUST NOT attempt to validate answers received in response to queries for the IPv6 AAAA address records for 'ipv4only.arpa', since, by definition, any such answers are generated by the local network's DNS64/NAT64 gateway, not the authoritative server responsible for that name.

4. For the purposes of this section, recursive resolvers fall into two categories. The first category is traditional recursive resolvers, which includes *forwarding* recursive resolvers, as commonly implemented in residential home gateways, and *iterative* recursive resolvers, as commonly deployed by ISPs. More information on these terms can be found in DNS Terminology [RFC8499]. The second category is DNS64 recursive resolvers, whose purpose is to synthesize IPv6 address records. These may be *forwarding* DNS64 recursive resolvers or *iterative* DNS64 recursive resolvers, and they work in partnership with a companion NAT64 gateway to communicate the appropriate NAT64 address synthesis prefix to clients.

   Traditional forwarding recursive resolvers SHOULD NOT recognize 'ipv4only.arpa' as special or give that name, or subdomains of that name, any special treatment. The rationale for this is that a traditional forwarding recursive resolver, such as built in to a residential home gateway, may itself be downstream of a DNS64 recursive resolver. Passing through the 'ipv4only.arpa' queries to the upstream DNS64 recursive resolver will allow the correct NAT64 prefix to be discovered.

   Traditional iterative recursive resolvers that are not explicitly configured to synthesize IPv6 prefixes on behalf of a companion NAT64 gateway need not recognize 'ipv4only.arpa' as special or take any special action.

   Forwarding or iterative recursive resolvers that have been explicitly configured to perform DNS64 address synthesis in support of a companion NAT64 gateway (i.e, "DNS64 recursive resolvers") MUST recognize 'ipv4only.arpa' as special. The authoritative name servers for 'ipv4only.arpa' cannot be expected to know the local network's NAT64 address synthesis prefix, so consulting the authoritative name servers for IPv6 address records for 'ipv4only.arpa' is futile. All DNS64 recursive resolvers MUST recognize 'ipv4only.arpa' (and all of its subdomains) as special, and MUST NOT attempt to look up NS records for 'ipv4only.arpa', or otherwise query authoritative name servers in an attempt to resolve this name. Instead, DNS64 recursive resolvers MUST act as authoritative for this zone, by generating immediate responses for all queries for 'ipv4only.arpa' (and any subdomain of 'ipv4only.arpa'), with the

one exception of queries for the DS record. Queries for the DS record are resolved the usual way to allow a client to securely verify that the 'ipv4only.arpa' zone has an insecure delegation. Note that this exception is not expected to receive widespread usage, since any client compliant with this specification already knows that 'ipv4only.arpa' is an insecure delegation and will not attempt DNSSEC validation for this name.

DNS64 recursive resolvers MUST generate the 192.0.0.170 and 192.0.0.171 responses for IPv4 address queries (DNS qtype "A"), the appropriate synthesized IPv6 address record responses for IPv6 address queries (DNS qtype "AAAA"), and a negative ("no error no answer") response for all other query types except DS.

For all subdomains of 'ipv4only.arpa', DNS64 recursive resolvers MUST generate immediate NXDOMAIN responses. All names falling below 'ipv4only.arpa' are defined to be nonexistent.

An example configuration for BIND 9 showing how to achieve the desired result is given in Appendix A.

Note that this is *not* a locally served zone in the usual sense of that term [RFC6303] because this rule applies *only* to DNS64 recursive resolvers, not to forwarding DNS recursive resolvers.

5. Authoritative name server software need not recognize 'ipv4only.arpa' as special or handle it in any special way.

6. Generally speaking, operators of authoritative name servers need not know anything about the name 'ipv4only.arpa', just as they do not need to know anything about any other names they are not responsible for. Only the administrators of the 'arpa' namespace need to be aware of this name's purpose and how it should be configured. In particular, 'ipv4only.arpa' MUST have the required records, and MUST be an insecure delegation, to allow DNS64 recursive resolvers to create synthesized AAAA answers within that zone. Making the 'ipv4only.arpa' zone a secure delegation would make it impossible for DNS64 recursive resolvers to create synthesized AAAA answers that will be accepted by DNSSEC validating clients, thereby defeating the entire purpose of the 'ipv4only.arpa' name.

7. DNS Registries/Registrars need not know anything about the name 'ipv4only.arpa', just as they do not need to know anything about any other name they are not responsible for.

7.2. Names ’170.0.0.192.in-addr.arpa’ and ’171.0.0.192.in-addr.arpa’

Since the IPv4 addresses 192.0.0.170 and 192.0.0.171 are defined to be special, and are listed in the IPv4 Special-Purpose Address Registry [SUv4], the corresponding reverse mapping names in the in-addr.arpa domain are similarly special.

The name ’170.0.0.192.in-addr.arpa’ is defined, by IETF specification [RFC7050], to have only one DNS record, type PTR, with rdata ’ipv4only.arpa’.

The name ’171.0.0.192.in-addr.arpa’ is defined, by IETF specification [RFC7050], to have only one DNS record, type PTR, with rdata ’ipv4only.arpa’.

There are no subdomains of ’170.0.0.192.in-addr.arpa’ or ’171.0.0.192.in-addr.arpa’. All names falling below these names are defined to be nonexistent (NXDOMAIN).

Practically speaking these two names are rarely used, but to the extent that they may be, they are special only to resolver APIs and libraries, as described in item 3 below:

1. Normal users should never have reason to encounter these two reverse mapping names. However, if they do, queries for these reverse mapping names should return the expected answer ’ipv4only.arpa’. Normal users have no need to know that these reverse mapping names are special.

2. Application software SHOULD NOT recognize these two reverse mapping names as special, and SHOULD NOT treat them differently. For example, if the user were to issue the Unix command "host 192.0.0.170" then the "host" command should call the name resolution API or library as usual and display the result that is returned.

3. Name resolution APIs and libraries SHOULD recognize these two reverse mapping names as special and generate the required responses locally. For the names ’170.0.0.192.in-addr.arpa’ and ’171.0.0.192.in-addr.arpa’ PTR queries yield the result ’ipv4only.arpa’; all other query types yield a negative ("no error no answer") response. For all subdomains of these two reverse mapping domains, all queries yield an NXDOMAIN response. All names falling below these two reverse mapping domains are defined to be nonexistent.

This local self-contained generation of these responses is to avoid placing unnecessary load on the authoritative 'in-addr.arpa' name servers.

4. Recursive resolvers SHOULD NOT recognize these two reverse mapping names as special and SHOULD NOT, by default, give them any special treatment.

5. Authoritative name server software need not recognize these two reverse mapping names as special or handle them in any special way.

6. Generally speaking, most operators of authoritative name servers need not know anything about these two reverse mapping names, just as they do not need to know anything about any other names they are not responsible for. Only the operators of the authoritative name servers for these two reverse mapping names need to be aware that these names are special, and require fixed answers specified by IETF specification.

7. DNS Registries/Registrars need not know anything about these two reverse mapping names, just as they do not need to know anything about any other name they are not responsible for.

7.2.1. ip6.arpa Reverse Mapping PTR Records

For all IPv6 addresses synthesized by a DNS64 recursive resolver, the DNS64 recursive resolver is responsible for synthesizing the appropriate 'ip6.arpa' reverse mapping PTR records too, if it chooses to provide reverse mapping PTR records. The same applies to the synthesized IPv6 addresses corresponding to the IPv4 addresses 192.0.0.170 and 192.0.0.171.

Generally a DNS64 recursive resolver synthesizes appropriate 'ip6.arpa' reverse mapping PTR records by extracting the embedded IPv4 address from the encoded IPv6 address, performing a reverse mapping PTR query for that IPv4 address, and then synthesizing a corresponding 'ip6.arpa' reverse mapping PTR record containing the same rdata.

In the case of synthesized IPv6 addresses corresponding to the IPv4 addresses 192.0.0.170 and 192.0.0.171, the DNS64 recursive resolver does not issue reverse mapping queries for those IPv4 addresses, but instead, according to rule 3 above, immediately returns the answer 'ipv4only.arpa'.

In the case of a client that uses the 'ipv4only.arpa' query to discover the IPv6 prefixes in use by the local NAT64 gateway, and then proceeds to perform its own address synthesis locally (which has benefits such as allowing DNSSEC validation), that client MUST also synthesize 'ip6.arpa' reverse mapping PTR records for those discovered prefix(es), according to the rules above: When a client's name resolution APIs and libraries receive a request to look up an 'ip6.arpa' reverse mapping PTR record for an address that falls within one of the discovered NAT64 address synthesis prefixes, the software extracts the embedded IPv4 address and then, for IPv4 addresses 192.0.0.170 and 192.0.0.171, returns the fixed answer 'ipv4only.arpa', and for all other IPv4 addresses performs a reverse mapping PTR query for the IPv4 address, and then synthesizes a corresponding 'ip6.arpa' reverse mapping PTR record containing the same rdata.

8. Acknowledgements

Thanks to Jouni Korhonen, Teemu Savolainen, and Dan Wing, for devising the NAT64 Prefix Discovery mechanism [RFC7050], and for their feedback on this document.

Thanks to Geoff Huston for his feedback on this document.

Thanks to Erik Kline for pointing out that the in-addr.arpa names are special too.

Thanks to Mark Andrews for conclusively pointing out the reasons why the 'ipv4only.arpa' zone must be an insecure delegation in order for the NAT64 Prefix Discovery mechanism [RFC7050] to work, and many other very helpful comments.

Thanks particularly to Lorenzo Colitti for an especially spirited hallway discussion at IETF 96 in Berlin, which lead directly to significant improvements in how this document presents the issues.

Thanks to Scott Bradner, Bernie Volz, Barry Leiba, Mirja Kuehlewind, Suresh Krishnan, Benjamin Kaduk, Roman Danyliw, Eric Vyncke and the other IESG reviewers for their thoughtful feedback.

Thanks to Dave Thaler and Warren Kumari for generously helping shepherd this document through the publication process.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <https://www.rfc-editor.org/info/rfc3646>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <https://www.rfc-editor.org/info/rfc6146>.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <https://www.rfc-editor.org/info/rfc6147>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <https://www.rfc-editor.org/info/rfc6761>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <https://www.rfc-editor.org/info/rfc7050>.

[RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration", RFC 8106, DOI 10.17487/RFC8106, March 2017, <https://www.rfc-editor.org/info/rfc8106>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, DOI 10.17487/RFC6303, July 2011, <https://www.rfc-editor.org/info/rfc6303>.

[RFC8244] Lemon, T., Droms, R., and W. Kumari, "Special-Use Domain Names Problem Statement", RFC 8244, DOI 10.17487/RFC8244, October 2017, <https://www.rfc-editor.org/info/rfc8244>.

[RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <https://www.rfc-editor.org/info/rfc8499>.

[SUDN] "Special-Use Domain Names Registry", <https://www.iana.org/assignments/special-use-domain-names/>.

[SUv4] "IANA IPv4 Special-Purpose Address Registry", <https://www.iana.org/assignments/iana-ipv4-special-registry/>.

[DNS1] "1.1.1.1 - The free app that makes your Internet safer", <https://1.1.1.1/>.

[DNS8] "Google Public DNS", <https://developers.google.com/speed/public-dns/>.

[DNS9] "Quad9 - Internet Security and Privacy In a Few Easy Steps", <https://quad9.net/>.

Appendix A. Example BIND 9 Configuration

A BIND 9 recursive resolver can be configured to act as authoritative for the necessary DNS64 names as described below.

In /etc/named.conf the following line is added:

```
zone "ipv4only.arpa"            { type master; file "ipv4only"; };
```

The file /var/named/ipv4only is created with the following content:

```
$TTL 86400              ; Default TTL 24 hours
@ IN SOA nameserver.example. admin.nameserver.example. (
         2016052400     ; Serial
         7200           ; Refresh ( 7200 = 2 hours)
         3600           ; Retry   ( 3600 = 1 hour)
         15724800       ; Expire  (15724800 = 6 months)
         60             ; Minimum
         )
@ IN NS  nameserver.example.

@ IN A    192.0.0.170
@ IN A    192.0.0.171
@ IN AAAA 64:ff9b::192.0.0.170 ; If not using Well-Known Prefix
@ IN AAAA 64:ff9b::192.0.0.171 ; place chosen NAT64 prefix here
```

Authors' Addresses

Stuart Cheshire
Apple Inc.
One Apple Park Way
Cupertino, California 95014
USA

Phone: +1 (408) 996-1010
Email: cheshire@apple.com

David Schinazi
Google LLC
1600 Amphitheatre Parkway
Mountain View, California 94043
USA

Email: dschinazi.ietf@gmail.com

Network Working Group
Internet-Draft
Updates: 4035 (if approved)
Intended status: Standards Track
Expires: November 25, 2017

K. Fujiwara
JPRS
A. Kato
Keio/WIDE
W. Kumari
Google

May 24, 2017

Aggressive use of DNSSEC-validated Cache
draft-ietf-dnsop-nsec-aggressiveuse-10

Abstract

The DNS relies upon caching to scale; however, the cache lookup generally requires an exact match. This document specifies the use of NSEC/NSEC3 resource records to allow DNSSEC validating resolvers to generate negative answers within a range, and positive answers from wildcards. This increases performance / decreases latency, decreases resource utilization on both authoritative and recursive servers, and also increases privacy. It may also help increase resilience to certain DoS attacks in some circumstances.

This document updates RFC4035 by allowing validating resolvers to generate negative answers based upon NSEC/NSEC3 records and positive answers in the presence of wildcards.

[Ed note: Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. RFC Editor, please remove before publication. This document is being collaborated on in Github at: https://github.com/wkumari/draft-ietf-dnsop-nsec-aggressiveuse. The most recent version of the document, open issues, etc should all be available here. The authors (gratefully) accept pull requests.]

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 25, 2017.

Table of Contents

1. Introduction

A DNS negative cache exists, and is used to cache the fact that an RRset does not exist. This method of negative caching requires exact matching; this leads to unnecessary additional lookups, increases latency, leads to extra resource utilization on both authoritative and recursive servers, and decreases privacy by leaking queries.

This document updates RFC 4035 to allow resolvers to use NSEC/NSEC3 resource records to synthesize negative answers from the information they have in the cache. This allows validating resolvers to respond with a negative answer immediately if the name in question falls into a range expressed by a NSEC/NSEC3 resource record already in the cache. It also allows the synthesis of positive answers in the presence of wildcard records.

Aggressive Negative Caching was first proposed in Section 6 of DNSSEC Lookaside Validation (DLV) [RFC5074] in order to find covering NSEC records efficiently.

[RFC8020] and [I-D.vixie-dnsext-resimprove] propose steps to using NXDOMAIN information for more effective caching. This document takes this technique further.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Many of the specialized terms used in this document are defined in DNS Terminology [RFC7719].

The key words "Source of Synthesis" in this document are to be interpreted as described in [RFC4592].

3. Problem Statement

The DNS negative cache caches negative (non-existent) information, and requires an exact match in most instances [RFC2308].

Assume that the (DNSSEC signed) "example.com" zone contains:

```
albatross.example.com IN A 192.0.2.1
elephant.example.com  IN A 192.0.2.2
zebra.example.com     IN A 192.0.2.3
```

If a validating resolver receives a query for cat.example.com, it contacts its resolver (which may be itself) to query the example.com servers and will get back an NSEC record stating that there are no records (alphabetically) between albatross and elephant, or an NSEC3 record stating there is nothing between two hashed names. The resolver then knows that cat.example.com does not exist; however, it does not use the fact that the proof covers a range (albatross to elephant) to suppress queries for other labels that fall within this range. This means that if the validating resolver gets a query for ball.example.com (or dog.example.com) it will once again go off and query the example.com servers for these names.

Apart from wasting bandwidth, this also wastes resources on the recursive server (it needs to keep state for outstanding queries), wastes resources on the authoritative server (it has to answer additional questions), increases latency (the end user has to wait longer than necessary to get back an NXDOMAIN answer), can be used by attackers to cause a DoS (see additional resources), and also has privacy implications (e.g: typos leak out further than necessary).

Another example: assume that the (DNSSEC signed) "example.org" zone contains:

```
avocado.example.org   IN A 192.0.2.1
*.example.org         IN A 192.0.2.2
zucchini.example.org IN A 192.0.2.3
```

If a query is received for leek.example.org, the system contacts its resolver (which may be itself) to query the example.org servers and will get back an NSEC record stating that there are no records (alphabetically) between avocado and zucchini (or an NSEC3 record stating there is nothing between two hashed names), as well as an answer for leek.example.org, with the label count of the signature set to two (see [RFC7129], section 5.3 for more details).

If the validating resolver gets a query for banana.example.org it will once again go off and query the example.org servers for banana.example.org (even though it already has proof that there is a wildcard record) - just like above, this has privacy implications, wastes resources, can be used to contribute to a DoS, etc.

4. Background

DNSSEC [RFC4035] and [RFC5155] both provide "authenticated denial of existence"; this is a cryptographic proof that the queried for name does not exist or type does not exist. Proof that a name does not exist is accomplished by providing a (DNSSEC secured) record containing the names which appear alphabetically before and after the

queried for name. In the first example above, if the (DNSSEC validating) recursive server were to query for dog.example.com it would receive a (signed) NSEC record stating that there are no labels between "albatross" and "elephant" (or, for NSEC3, a similar pair of hashed names). This is a signed, cryptographic proof that these names are the ones before and after the queried for label. As dog.example.com falls within this range, the recursive server knows that dog.example.com really does not exist. Proof that a type does not exist is accomplished by providing a (DNSSEC secured) record containing the queried for name, and a type bitmap which does not include the requested type.

This document specifies that this NSEC/NSEC3 record should be used to generate negative answers for any queries that the validating server receives that fall within the range covered by the record (for the TTL for the record). This document also specifies that a positive answer should be generated for any queries that the validating server receives that are proven to be covered by a wildcard record.

Section 4.5 of [RFC4035] says:

"In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace." and "The reason for these recommendations is that, between the initial query and the expiration of the data from the cache, the authoritative data might have been changed (for example, via dynamic update).". In other words, if a resolver generates negative answers from an NSEC record, it will not send any queries for names within that NSEC range (for the TTL). If a new name is added to the zone during this interval the resolver will not know this. Similarly, if the resolver is generating responses from a wildcard record, it will continue to do so (for the TTL).

We believe this recommendation can be relaxed because, in the absence of this technique, a lookup for the exact name could have come in during this interval, and so a negative answer could already be cached (see [RFC2308] for more background). This means that zone operators should have no expectation that an added name would work immediately. With DNSSEC and Aggressive NSEC, the TTL of the NSEC/NSEC3 record and the SOA.MINIMUM field are the authoritative statement of how quickly a name can start working within a zone.

5. Aggressive use of Cache

This document relaxes the restriction given in Section 4.5 of [RFC4035], see Section 7 for more detail.

If the negative cache of the validating resolver has sufficient information to validate the query, the resolver SHOULD use NSEC, NSEC3 and wildcard records to synthesize answers as described in this document. Otherwise, it MUST fall back to send the query to the authoritative DNS servers.

5.1. NSEC

The validating resolver needs to check the existence of an NSEC RR matching/covering the source of synthesis and an NSEC RR covering the query name.

If denial of existence can be determined according to the rules set out in Section 5.4 of [RFC4035], using NSEC records in the cache, then the resolver can immediately return an NXDOMAIN or NODATA (as appropriate) response.

5.2. NSEC3

NSEC3 aggressive negative caching is more difficult than NSEC aggressive caching. If the zone is signed with NSEC3, the validating resolver needs to check the existence of non-terminals and wildcards which derive from query names.

If denial of existence can be determined according to the rules set out in [RFC5155] Sections 8.4, 8.5, 8.6, 8.7, using NSEC3 records in the cache, then the resolver can immediately return an NXDOMAIN or NODATA response (as appropriate).

If a covering NSEC3 RR has Opt-Out flag, the covering NSEC3 RR does not prove the non-existence of the domain name and the aggressive negative caching is not possible for the domain name.

5.3. Wildcards

The last paragraph of [RFC4035] Section 4.5 also discusses the use of wildcards and NSEC RRs to generate positive responses and recommends that it not be relied upon. Just like the case for the aggressive use of NSEC/NSEC3 for negative answers, we revise this recommendation.

As long as the validating resolver can determine that a name would not exist without the wildcard match, determined according to the

rules set out in Section 5.3.4 of [RFC4035] (NSEC), or in Section 8.8 of [RFC5155], it SHOULD synthesize an answer (or NODATA response) for that name using the cached deduced wildcard. If the corresponding wildcard record is not in the cache, it MUST fall back to send the query to the authoritative DNS servers.

5.4. Consideration on TTL

The TTL value of negative information is especially important, because newly added domain names cannot be used while the negative information is effective.

Section 5 of [RFC2308] suggests a maximum default negative cache TTL value of 3 hours (10800). It is RECOMMENDED that validating resolvers limit the maximum effective TTL value of negative responses (NSEC/NSEC3 RRs) to this same value.

Section 5 of [RFC2308] also states that a negative cache entry TTL is taken from the minimum of the SOA.MINIMUM field and SOA's TTL. This can be less than the TTL of an NSEC or NSEC3 record, since their TTL is equal to the SOA.MINIMUM field (see [RFC4035]section 2.3 and [RFC5155] section 3.)

A resolver that supports aggressive use of NSEC and NSEC3 SHOULD reduce the TTL of NSEC and NSEC3 records to match the SOA.MINIMUM field in the authority section of a negative response, if SOA.MINIMUM is smaller.

6. Benefits

The techniques described in this document provide a number of benefits, including (in no specific order):

Reduced latency: By answering directly from cache, validating resolvers can immediately inform clients that the name they are looking for does not exist, improving the user experience.

Decreased recursive server load: By answering queries from the cache by synthesizing answers, validating servers avoid having to send a query and wait for a response. In addition to decreasing the bandwidth used, it also means that the server does not need to allocate and maintain state, thereby decreasing memory and CPU load.

Decreased authoritative server load: Because recursive servers can answer queries without asking the authoritative server, the authoritative servers receive fewer queries. This decreases the

authoritative server bandwidth, queries per second and CPU utilization.

The scale of the benefit depends upon multiple factors, including the query distribution. For example, at the time of this writing, around 65% of queries to Root Name servers result in NXDOMAIN responses (see statistics from [root-servers.org]); this technique will eliminate a sizable quantity of these.

The technique described in this document may also mitigate so-called "random QNAME attacks", in which attackers send many queries for random sub-domains to resolvers. As the resolver will not have the answers cached, it has to ask external servers for each random query, leading to a DoS on the authoritative servers (and often resolvers). Aggressive NSEC may help mitigate these attacks by allowing the resolver to answer directly from cache for any random queries which fall within already requested ranges. It will not always work as an effective defense, not least because not many zones are DNSSEC signed at all -- but it will still provide an additional layer of defense.

As these benefits are only accrued by those using DNSSEC, it is hoped that these techniques will lead to more DNSSEC deployment.

7. Update to RFC 4035

Section 4.5 of [RFC4035] shows that "In theory, a resolver could use wildcards or NSEC RRs to generate positive and negative responses (respectively) until the TTL or signatures on the records in question expire. However, it seems prudent for resolvers to avoid blocking new authoritative data or synthesizing new data on their own. Resolvers that follow this recommendation will have a more consistent view of the namespace".

The paragraph is updated as follows:

```
+------------------------------------------------------------------+
|  Once the records are validated, DNSSEC enabled validating       |
|  resolvers SHOULD use wildcards and NSEC/NSEC3 resource records  |
|  to generate positive and negative responses until the           |
|  effective TTLs or signatures for those records expire.          |
+------------------------------------------------------------------+
```

8. IANA Considerations

This document has no IANA actions.

9. Security Considerations

Use of NSEC / NSEC3 resource records without DNSSEC validation may create serious security issues, and so this technique requires DNSSEC validation.

Newly registered resource records may not be used immediately. However, choosing suitable TTL value and negative cache TTL value (SOA MINIMUM field) will mitigate the delay concern, and it is not a security problem.

It is also suggested to limit the maximum TTL value of NSEC / NSEC3 resource records in the negative cache to, for example, 10800 seconds (3hrs), to mitigate this issue.

Although the TTL of NSEC/NSEC3 records is typically fairly short (minutes or hours), their RRSIG expiration time can be much further in the future (weeks). An attacker who is able to successfully spoof responses might poison a cache with old NSEC/NSEC3 records. If the resolver is not making aggressive use of NSEC/NSEC3, the attacker has to repeat the attack for every query. If the resolver is making aggressive use of NSEC/NSEC3, one successful attack would be able to suppress many queries for new names, up to the negative TTL.

10. Implementation Status

[Editor note: RFC Editor, please remove this entire section. RFC6982 says: "Since this information is necessarily time dependent, it is inappropriate for inclusion in a published RFC."]

Unbound currently implements aggressive negative caching, as does Google Public DNS.

11. Acknowledgments

The authors gratefully acknowledge DLV [RFC5074] author Samuel Weiler and the Unbound developers.

Thanks to Mark Andrews for providing the helpful notes for implementors provided in Appendix B.

The authors would like to specifically thank Stephane Bortzmeyer (for standing next to and helping edit), Ralph Dolmans, Tony Finch, Tatuya JINMEI for extensive review and comments, and also Mark Andrews, Casey Deccio, Alexander Dupuy, Olafur Gudmundsson, Bob Harold, Shumon Huque, John Levine, Pieter Lexis, Matthijs Mekking (who even sent pull requests!) and Ondrej Sury.

11.1. Change History

RFC Editor: Please remove this section prior to publication.

-09 to -10:

- Addressed IESG comments at https://datatracker.ietf.org/doc/draft-ietf-dnsop-nsec-aggressiveuse/ballot/
- Main change "the resolver SHOULD use NSEC, NSEC3 and wildcard records aggressively." -> "HOULD use NSEC, NSEC3 and wildcard records to synthesize answers as described in this document" (Mirja) - aggressively wasn’t really described...

-08 to -09:

- Made RFC5074 Informative (after discussions with chairs.
- Addressed SecDir comments.
- Addressed OpsDir comments.

-06 to -08:

- Largely editorial, but please see the diffs (editors forgot to update change log when editing, backfilling change log.)
- Changed "replacement" text to be "DNSSEC enabled validating resolvers SHOULD use wildcards ..." to align with text in doc.
- "A resolver that supports aggressive use of NSEC and NSEC3 SHOULD" (should -> SHOULD) - to align with rest of text.

-05 to -06:

- Moved some dangling text around - when the examples were added some text added in the wrong place.
- There were some bits which mentioned "negative" in the title.
- We had the cut-and-paste of what changed in 4035 twice.
- Clarified that this also allows NODATA responses to be synthesized.

-04 to -05:

- Bob pointed out that I did a stupid - when I added the wildcard to ’example.com’ I made the example wrong / confusing. I have attempted to fix this by adding a second example zone (example.org) with the wildcard instead.

- More helpful changes (in a pull request, thanks!) from Matthijs

- Included Mark Andrew’s useful explanation of how to tell ENT from NXD as an Appendix.

-03 to -04:

- Working group does want the "positive" answers, not just negative ones. This requires reading what used to be Section 7, and a bunch of cleanup, including:

  * Additional text in the Problem Statement

  * Added a wildcard record to the zone.

  * Added "or positive answers from wildcards" type text (where appropriate) to explain that this isn’t just for negative answers.

  * Reworded much of the Wildcard text.

- Incorporated pull request from Tony Finch (thanks!): https://github.com/wkumari/draft-ietf-dnsop-nsec-aggressiveuse/pull/1

- More fixups from Tony (including text): https://www.ietf.org/mail-archive/web/dnsop/current/msg18271.html. This included much clearer text on TTL, references to the NSEC / NSEC3 RFCs (instead of my clumsy summary), good text on replays, etc.

- Converted the "zone file" to a figure to make it more readable.

- Text from Tim W: "If a validating resolver receives a query for cat.example.com, it contacts its resolver (which may be itself) to query..." - which satisfies Jinmei’s concern (which I was too dense to grock).

- Fixup of the "validation required" in security considerations.

-02 to -03:

- Integrated a bunch of comments from Matthijs Mekking - details in: https://github.com/wkumari/draft-ietf-dnsop-nsec-aggressiveuse/

pull/1. I decided to keep "Aggressive Negative Caching" instead of "Aggressive USE OF Negative Caching" for readability.

- Attempted to address Bob Harold's comment on the readability issues with "But, it will be more effective when both are enabled..." in Section 5.4 - https://www.ietf.org/mail-archive/web/dnsop/current/msg17997.html

- MAYs and SHOULD drifted in the text block. Fixed - thanks to https://mailarchive.ietf.org/arch/msg/dnsop/2ljmmzxtIMCFMLOZmWcSbTYVOy4

- A number of good edits from Stephane in: https://www.ietf.org/mail-archive/web/dnsop/current/msg18109.html

- A bunch more edits from Jinmei, as in: https://www.ietf.org/mail-archive/web/dnsop/current/msg18206.html

-01 to -02:

- Added Section 6 - Benefits (as suggested by Jinmei).

- Removed Appendix B (Jinmei)

- Replaced "full-service" with "validating" (where applicable)

- Integrated other comments from Jinmei from https://www.ietf.org/mail-archive/web/dnsop/current/msg17875.html

- Integrated comment from co-authors, including re-adding parts of Appendix B, terminology, typos.

- Tried to explain under what conditions this may actually mitigate attacks.

-00 to -01:

- Comments from DNSOP meeting in Berlin.

- Changed intended status to Standards Track (updates RFC 4035)

- Added a section "Updates to RFC 4035"

- Some language clarification / typo / cleanup

- Cleaned up the TTL section a bit.

- Removed Effects section, Additional proposal section, and pseudo code.
- Moved "mitigation of random subdomain attacks" to Appendix.

From draft-fujiwara-dnsop-nsec-aggressiveuse-03 -> draft-ietf-dnsop-nsec-aggressiveuse

- Document adopted by DNSOP WG.
- Adoption comments
- Changed main purpose to performance
- Use NSEC3/Wildcard keywords
- Improved wordings (from good comments)
- Simplified pseudo code for NSEC3
- Added Warren as co-author.
- Reworded much of the problem statement
- Reworked examples to better explain the problem / solution.

11.1.1. Version draft-fujiwara-dnsop-nsec-aggressiveuse-01

- Added reference to DLV [RFC5074] and imported some sentences.
- Added Aggressive Negative Caching Flag idea.
- Added detailed algorithms.

11.1.2. Version draft-fujiwara-dnsop-nsec-aggressiveuse-02

- Added reference to [I-D.vixie-dnsext-resimprove]
- Added considerations for the CD bit
- Updated detailed algorithms.
- Moved Aggressive Negative Caching Flag idea into Additional Proposals

11.1.3. Version draft-fujiwara-dnsop-nsec-aggressiveuse-03

- Added "Partial implementation"
- Section 4,5,6 reorganized for better representation
- Added NODATA answer in Section 4
- Trivial updates
- Updated pseudo code

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <http://www.rfc-editor.org/info/rfc2308>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <http://www.rfc-editor.org/info/rfc4035>.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <http://www.rfc-editor.org/info/rfc4592>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <http://www.rfc-editor.org/info/rfc5155>.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <http://www.rfc-editor.org/info/rfc7129>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <http://www.rfc-editor.org/info/rfc7719>.

12.2. Informative References

[I-D.vixie-dnsext-resimprove]
Vixie, P., Joffe, R., and F. Neves, "Improvements to DNS Resolvers for Resiliency, Robustness, and Responsiveness", draft-vixie-dnsext-resimprove-00 (work in progress), June 2010.

[RFC5074] Weiler, S., "DNSSEC Lookaside Validation (DLV)", RFC 5074, DOI 10.17487/RFC5074, November 2007, <http://www.rfc-editor.org/info/rfc5074>.

[RFC8020] Bortzmeyer, S. and S. Huque, "NXDOMAIN: There Really Is Nothing Underneath", RFC 8020, DOI 10.17487/RFC8020, November 2016, <http://www.rfc-editor.org/info/rfc8020>.

[root-servers.org]
IANA, "Root Server Technical Operations Assn", <http://www.root-servers.org/>.

Appendix A. Detailed implementation notes

- Previously, cached negative responses were indexed by QNAME, QCLASS, QTYPE, and the setting of the CD bit (see RFC 4035, Section 4.7), and only queries matching the index key would be answered from the cache. With aggressive negative caching, the validator, in addition to checking to see if the answer is in its cache before sending a query, checks to see whether any cached and validated NSEC record denies the existence of the sought record(s). Using aggressive negative caching, a validator will not make queries for any name covered by a cached and validated NSEC record. Furthermore, a validator answering queries from clients will synthesize a negative answer (or NODATA response) whenever it has an applicable validated NSEC in its cache unless the CD bit was set on the incoming query. (Imported from Section 6 of [RFC5074]).

- Implementing aggressive negative caching suggests that a validator will need to build an ordered data structure of NSEC and NSEC3 records for each signer domain name of NSEC / NSEC3 records in order to efficiently find covering NSEC / NSEC3 records. Call the table as NSEC_TABLE. (Imported from Section 6.1 of [RFC5074] and expanded.)

- The aggressive negative caching may be inserted at the cache lookup part of the recursive resolvers.

- If errors happen in aggressive negative caching algorithm, resolvers MUST fall back to resolve the query as usual. "Resolve the query as usual" means that the resolver must process the query as though it does not implement aggressive negative caching.

Appendix B. Procedure for determining ENT vs NXDOMAN with NSEC

This procedure outlines how to determine if a given name does not exist, or is an ENT (Empty Non-Terminal, see [RFC5155] Section 1.3) with NSEC.

If the NSEC record has not been verified as secure discard it.

If the given name sorts before or matches the NSEC owner name discard it as it does not prove the NXDOMAIN or ENT.

If the given name is a subdomain of the NSEC owner name and the NS bit is present and the SOA bit is absent then discard the NSEC as it is from a parent zone.

If the next domain name sorts after the NSEC owner name and the given name sorts after or matches next domain name then discard the NSEC record as it does not prove the NXDOMAIN or ENT.

If the next domain name sorts before or matches the NSEC owner name and the given name is not a subdomain of the next domain name then discard the NSEC as it does not prove the NXDOMAIN or ENT.

You now have a NSEC record that proves the NXDOMAIN or ENT.

If the next domain name is a subdomain of the given name you have a ENT otherwise you have a NXDOMAIN.

Authors' Addresses

Kazunori Fujiwara
Japan Registry Services Co., Ltd.
Chiyoda First Bldg. East 13F, 3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: fujiwara@jprs.co.jp

Akira Kato
Keio University/WIDE Project
Graduate School of Media Design, 4-1-1 Hiyoshi
Kohoku, Yokohama 223-8526
Japan

Phone: +81 45 564 2490
Email: kato@wide.ad.jp

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Network Working Group
Internet-Draft
Obsoletes: 7719 (if approved)
Updates: 2308 (if approved)
Intended status: Best Current Practice
Expires: March 17, 2019

P. Hoffman
ICANN
A. Sullivan

K. Fujiwara
JPRS

September 13, 2018

DNS Terminology
draft-ietf-dnsop-terminology-bis-14

Abstract

The domain name system (DNS) is defined in literally dozens of different RFCs. The terminology used by implementers and developers of DNS protocols, and by operators of DNS systems, has sometimes changed in the decades since the DNS was first defined. This document gives current definitions for many of the terms used in the DNS in a single document.

This document obsoletes RFC 7719 and updates RFC 2308.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 17, 2019.

Table of Contents

1. Introduction

The Domain Name System (DNS) is a simple query-response protocol whose messages in both directions have the same format. (Section 2 gives a definition of "public DNS", which is often what people mean when they say "the DNS".) The protocol and message format are defined in [RFC1034] and [RFC1035]. These RFCs defined some terms, but later documents defined others. Some of the terms from [RFC1034] and [RFC1035] now have somewhat different meanings than they did in 1987.

This document collects a wide variety of DNS-related terms. Some of them have been precisely defined in earlier RFCs, some have been loosely defined in earlier RFCs, and some are not defined in any earlier RFC at all.

Most of the definitions here are the consensus definition of the DNS community -- both protocol developers and operators. Some of the definitions differ from earlier RFCs, and those differences are noted. In this document, where the consensus definition is the same as the one in an RFC, that RFC is quoted. Where the consensus definition has changed somewhat, the RFC is mentioned but the new stand-alone definition is given. See Appendix A for a list of the definitions that this document updates.

It is important to note that, during the development of this document, it became clear that some DNS-related terms are interpreted quite differently by different DNS experts. Further, some terms that are defined in early DNS RFCs now have definitions that are generally agreed to, but that are different from the original definitions. Therefore, this document is a substantial revision to [RFC7719].

The terms are organized loosely by topic. Some definitions are for new terms for things that are commonly talked about in the DNS community but that never had terms defined for them.

Other organizations sometimes define DNS-related terms their own way. For example, the WHATWG defines "domain" at <https://url.spec.whatwg.org/>. The Root Server System Advisory Committee (RSSAC) has a good lexicon [RSSAC026].

Note that there is no single consistent definition of "the DNS". It can be considered to be some combination of the following: a commonly used naming scheme for objects on the Internet; a distributed database representing the names and certain properties of these objects; an architecture providing distributed maintenance, resilience, and loose coherency for this database; and a simple query-response protocol (as mentioned below) implementing this architecture. Section 2 defines "global DNS" and "private DNS" as a way to deal with these differing definitions.

Capitalization in DNS terms is often inconsistent among RFCs and various DNS practitioners. The capitalization used in this document is a best guess at current practices, and is not meant to indicate that other capitalization styles are wrong or archaic. In some cases, multiple styles of capitalization are used for the same term due to quoting from different RFCs.

Readers should note that the terms in this document are grouped by topic. Someone who is not already familiar with the DNS can probably not learn about the DNS from scratch by reading this document from front to back. Instead, skipping around may be the only way to get enough context to understand some of the definitions. This document

has an index that might be useful for readers who are attempting to learn the DNS by reading this document.

2. Names

Naming system: A naming system associates names with data. Naming systems have many significant facets that help differentiate them from each other. Some commonly-identified facets include:

* Composition of names
* Format of names
* Administration of names
* Types of data that can be associated with names
* Types of metadata for names
* Protocol for getting data from a name
* Context for resolving a name

Note that this list is a small subset of facets that people have identified over time for naming systems, and the IETF has yet to agree on a good set of facets that can be used to compare naming systems. For example, other facets might include "protocol to update data in a name", "privacy of names", and "privacy of data associated with names", but those are not as well-defined as the ones listed above. The list here is chosen because it helps describe the DNS and naming systems similar to the DNS.

Domain name: An ordered list of one or more labels.

Note that this is a definition independent of the DNS RFCs, and the definition here also applies to systems other than the DNS. [RFC1034] defines the "domain name space" using mathematical trees and their nodes in graph theory, and this definition has the same practical result as the definition here. Any path of a directed acyclic graph can be represented by a domain name consisting of the labels of its nodes, ordered by decreasing distance from the root(s) (which is the normal convention within the DNS, including this document). A domain name whose last label identifies a root of the graph is fully qualified; other domain names whose labels form a strict prefix of a fully qualified domain name are relative to its first omitted node.

Also note that different IETF and non-IETF documents have used the term "domain name" in many different ways. It is common for earlier documents to use "domain name" to mean "names that match the syntax in [RFC1035]", but possibly with additional rules such as "and are, or will be, resolvable in the global DNS" or "but only using the presentation format".

Label: An ordered list of zero or more octets that makes up a portion of a domain name. Using graph theory, a label identifies one node in a portion of the graph of all possible domain names.

Global DNS: Using the short set of facets listed in "Naming system", the global DNS can be defined as follows. Most of the rules here come from [RFC1034] and [RFC1035], although the term "global DNS" has not been defined before now.

Composition of names -- A name in the global DNS has one or more labels. The length of each label is between 0 and 63 octets inclusive. In a fully-qualified domain name, the last label in the ordered list is 0 octets long; it is the only label whose length may be 0 octets, and it is called the "root" or "root label". A domain name in the global DNS has a maximum total length of 255 octets in the wire format; the root represents one octet for this calculation. (Multicast DNS [RFC6762] allows names up to 255 bytes plus a terminating zero byte based on a different interpretation of RFC 1035 and what is included in the 255 octets.)

Format of names -- Names in the global DNS are domain names. There are three formats: wire format, presentation format, and common display.

The basic wire format for names in the global DNS is a list of labels ordered by decreasing distance from the root, with the root label last. Each label is preceded by a length octet. [RFC1035] also defines a compression scheme that modifies this format.

The presentation format for names in the global DNS is a list of labels ordered by decreasing distance from the root, encoded as ASCII, with a "." character between each label. In presentation format, a fully-qualified domain name includes the root label and the associated separator dot. For example, in presentation format, a fully-qualified domain name with two non-root labels is always shown as "example.tld." instead of "example.tld". [RFC1035] defines a method for showing octets that do not display in ASCII.

The common display format is used in applications and free text. It is the same as the presentation format, but showing the root label and the "." before it is optional and is rarely done. For example, in common display format, a fully-qualified domain name with two non-root labels is usually shown as "example.tld" instead of "example.tld.". Names in the common display format are normally written such that the directionality of the writing system presents labels by decreasing distance from the root (so, in both English and the C programming language the root or TLD label in the ordered list is right-most; but in Arabic it may be left-most, depending on local conventions).

Administration of names -- Administration is specified by delegation (see the definition of "delegation" in Section 7). Policies for administration of the root zone in the global DNS are determined by the names operational community, which convenes itself in the Internet Corporation for Assigned Names and Numbers (ICANN). The names operational community selects the IANA Functions Operator for the global DNS root zone. At the time this document is published, that operator is Public Technical Identifiers (PTI). (See <https://pti.icann.org/> for more information about PTI operating the IANA Functions.) The name servers that serve the root zone are provided by independent root operators. Other zones in the global DNS have their own policies for administration.

Types of data that can be associated with names -- A name can have zero or more resource records associated with it. There are numerous types of resource records with unique data structures defined in many different RFCs and in the IANA registry at [IANA_Resource_Registry].

Types of metadata for names -- Any name that is published in the DNS appears as a set of resource records (see the definition of "RRset" in Section 5). Some names do not themselves have data associated with them in the DNS, but "appear" in the DNS anyway because they form part of a longer name that does have data associated with it (see the definition of "empty non-terminals" in Section 7).

Protocol for getting data from a name -- The protocol described in [RFC1035].

Context for resolving a name -- The global DNS root zone distributed by PTI.

Private DNS: Names that use the protocol described in [RFC1035] but that do not rely on the global DNS root zone, or names that are

otherwise not generally available on the Internet but are using the protocol described in [RFC1035]. A system can use both the global DNS and one or more private DNS systems; for example, see "Split DNS" in Section 6.

Note that domain names that do not appear in the DNS, and that are intended never to be looked up using the DNS protocol, are not part of the global DNS or a private DNS even though they are domain names.

Multicast DNS: "Multicast DNS (mDNS) provides the ability to perform DNS-like operations on the local link in the absence of any conventional Unicast DNS server. In addition, Multicast DNS designates a portion of the DNS namespace to be free for local use, without the need to pay any annual fee, and without the need to set up delegations or otherwise configure a conventional DNS server to answer for those names." (Quoted from [RFC6762], Abstract) Although it uses a compatible wire format, mDNS is strictly speaking a different protocol than DNS. Also, where the above quote says "a portion of the DNS namespace", it would be clearer to say "a portion of the domain name space" The names in mDNS are not intended to be looked up in the DNS.

Locally served DNS zone: A locally served DNS zone is a special case of private DNS. Names are resolved using the DNS protocol in a local context. [RFC6303] defines subdomains of IN-ADDR.ARPA that are locally served zones. Resolution of names through locally served zones may result in ambiguous results. For example, the same name may resolve to different results in different locally served DNS zone contexts. The context for a locally served DNS zone may be explicit, for example, as defined in [RFC6303], or implicit, as defined by local DNS administration and not known to the resolution client.

Fully qualified domain name (FQDN): This is often just a clear way of saying the same thing as "domain name of a node", as outlined above. However, the term is ambiguous. Strictly speaking, a fully qualified domain name would include every label, including the zero-length label of the root: such a name would be written "www.example.net." (note the terminating dot). But because every name eventually shares the common root, names are often written relative to the root (such as "www.example.net") and are still called "fully qualified". This term first appeared in [RFC0819]. In this document, names are often written relative to the root.

The need for the term "fully qualified domain name" comes from the existence of partially qualified domain names, which are names where one or more of the last labels in the ordered list are

omitted (for example, a domain name of "www" relative to "example.net" identifies "www.example.net"). Such relative names are understood only by context.

Host name: This term and its equivalent, "hostname", have been widely used but are not defined in [RFC1034], [RFC1035], [RFC1123], or [RFC2181]. The DNS was originally deployed into the Host Tables environment as outlined in [RFC0952], and it is likely that the term followed informally from the definition there. Over time, the definition seems to have shifted. "Host name" is often meant to be a domain name that follows the rules in Section 3.5 of [RFC1034], the "preferred name syntax" (that is, every character in each label is a letter, a digit, or a hyphen). Note that any label in a domain name can contain any octet value; hostnames are generally considered to be domain names where every label follows the rules in the "preferred name syntax", with the amendment that labels can start with ASCII digits (this amendment comes from Section 2.1 of [RFC1123]).

People also sometimes use the term hostname to refer to just the first label of an FQDN, such as "printer" in "printer.admin.example.com". (Sometimes this is formalized in configuration in operating systems.) In addition, people sometimes use this term to describe any name that refers to a machine, and those might include labels that do not conform to the "preferred name syntax".

TLD: A Top-Level Domain, meaning a zone that is one layer below the root, such as "com" or "jp". There is nothing special, from the point of view of the DNS, about TLDs. Most of them are also delegation-centric zones (defined in Section 7, and there are significant policy issues around their operation. TLDs are often divided into sub-groups such as Country Code Top-Level Domains (ccTLDs), Generic Top-Level Domains (gTLDs), and others; the division is a matter of policy, and beyond the scope of this document.

IDN: The common abbreviation for "Internationalized Domain Name". The IDNA protocol is the standard mechanism for handling domain names with non-ASCII characters in applications in the DNS. The current standard at the time of this writing, normally called "IDNA2008", is defined in [RFC5890], [RFC5891], [RFC5892], [RFC5893], and [RFC5894]. These documents define many IDN-specific terms such as "LDH label", "A-label", and "U-label". [RFC6365] defines more terms that relate to internationalization (some of which relate to IDNs), and [RFC6055] has a much more extensive discussion of IDNs, including some new terminology.

Subdomain: "A domain is a subdomain of another domain if it is contained within that domain. This relationship can be tested by seeing if the subdomain's name ends with the containing domain's name." (Quoted from [RFC1034], Section 3.1). For example, in the host name "nnn.mmm.example.com", both "mmm.example.com" and "nnn.mmm.example.com" are subdomains of "example.com". Note that the comparisons here are done on whole labels; that is, "ooo.example.com" is not a subdomain of "oo.example.com".

Alias: The owner of a CNAME resource record, or a subdomain of the owner of a DNAME resource record (DNAME records are defined in [RFC6672]). See also "canonical name".

Canonical name: A CNAME resource record "identifies its owner name as an alias, and specifies the corresponding canonical name in the RDATA section of the RR." (Quoted from [RFC1034], Section 3.6.2) This usage of the word "canonical" is related to the mathematical concept of "canonical form".

CNAME: "It is traditional to refer to the owner of a CNAME record as 'a CNAME'. This is unfortunate, as 'CNAME' is an abbreviation of 'canonical name', and the owner of a CNAME record is an alias, not a canonical name." (Quoted from [RFC2181], Section 10.1.1)

3. DNS Response Codes

Some of response codes that are defined in [RFC1035] have acquired their own shorthand names. All of the RCODEs are listed at [IANA_Resource_Registry], although that site uses mixed-case capitalization, while most documents use all-caps. Some of the common names are described here, but the official list is in the IANA registry.

NOERROR: "No error condition" (Quoted from [RFC1035], Section 4.1.1.)

FORMERR: "Format error - The name server was unable to interpret the query." (Quoted from [RFC1035], Section 4.1.1.)

SERVFAIL: "Server failure - The name server was unable to process this query due to a problem with the name server." (Quoted from [RFC1035], Section 4.1.1.)

NXDOMAIN: "Name Error - This code signifies that the domain name referenced in the query does not exist." (Quoted from [RFC1035], Section 4.1.1.) [RFC2308] established NXDOMAIN as a synonym for Name Error.

NOTIMP: "Not Implemented - The name server does not support the requested kind of query." (Quoted from [RFC1035], Section 4.1.1.)

REFUSED: "Refused - The name server refuses to perform the specified operation for policy reasons. For example, a name server may not wish to provide the information to the particular requester, or a name server may not wish to perform a particular operation (e.g., zone transfer) for particular data." (Quoted from [RFC1035], Section 4.1.1.)

NODATA: "A pseudo RCODE which indicates that the name is valid for the given class, but there are no records of the given type. A NODATA response has to be inferred from the answer." (Quoted from [RFC2308], Section 1.) "NODATA is indicated by an answer with the RCODE set to NOERROR and no relevant answers in the answer section. The authority section will contain an SOA record, or there will be no NS records there." (Quoted from [RFC2308], Section 2.2.) Note that referrals have a similar format to NODATA replies; [RFC2308] explains how to distinguish them.

The term "NXRRSET" is sometimes used as a synonym for NODATA. However, this is a mistake, given that NXRRSET is a specific error code defined in [RFC2136].

Negative response: A response that indicates that a particular RRset does not exist, or whose RCODE indicates the nameserver cannot answer. Sections 2 and 7 of [RFC2308] describe the types of negative responses in detail.

4. DNS Transactions

The header of a DNS message is its first 12 octets. Many of the fields and flags in the header diagram in Sections 4.1.1 through 4.1.3 of [RFC1035] are referred to by their names in that diagram. For example, the response codes are called "RCODEs", the data for a record is called the "RDATA", and the authoritative answer bit is often called "the AA flag" or "the AA bit".

Class: A class "identifies a protocol family or instance of a protocol" (Quoted from [RFC1034], Section 3.6). "The DNS tags all data with a class as well as the type, so that we can allow parallel use of different formats for data of type address." (Quoted from [RFC1034], Section 2.2). In practice, the class for nearly every query is "IN". There are some queries for "CH", but they are usually for the purposes of information about the server itself rather than for a different type of address.

QNAME: The most commonly-used rough definition is that the QNAME is a field in the Question section of a query. "A standard query specifies a target domain name (QNAME), query type (QTYPE), and query class (QCLASS) and asks for RRs which match." (Quoted from [RFC1034], Section 3.7.1.). Strictly speaking, the definition comes from [RFC1035], Section 4.1.2, where the QNAME is defined in respect of the Question Section. This definition appears to be applied consistently: the discussion of inverse queries in section 6.4 refers to the "owner name of the query RR and its TTL", because inverse queries populate the Answer Section and leave the Question Section empty. (Inverse queries are deprecated in [RFC3425], and so relevant definitions do not appear in this document.)

[RFC2308], however, has an alternate definition that puts the QNAME in the answer (or series of answers) instead of the query. It defines QNAME as: "...the name in the query section of an answer, or where this resolves to a CNAME, or CNAME chain, the data field of the last CNAME. The last CNAME in this sense is that which contains a value which does not resolve to another CNAME." This definition has a certain internal logic, because of the way CNAME substitution works and the definition of CNAME. If a name server does not find an RRset that matches a query, but it finds the same name in the same class with a CNAME record, then the name server "includes the CNAME record in the response and restarts the query at the domain name specified in the data field of the CNAME record." (Quoted from [RFC1034] Section 3.6.2). This is made explicit in the resolution algorithm outlined in Section 4.3.2 of [RFC1034], which says to "change QNAME to the canonical name in the CNAME RR, and go back to step 1" in the case of a CNAME RR. Since a CNAME record explicitly declares that the owner name is canonically named what is in the RDATA, then there is a way to view the new name (i.e. the name that was in the RDATA of the CNAME RR) as also being the QNAME.

This creates a kind of confusion, however, because the response to a query that results in CNAME processing contains in the echoed Question Section one QNAME (the name in the original query), and a second QNAME that is in the data field of the last CNAME. The confusion comes from the iterative/recursive mode of resolution, which finally returns an answer that need not actually have the same owner name as the QNAME contained in the original query.

To address this potential confusion, it is helpful to distinguish between three meanings:

* QNAME (original): The name actually sent in the Question Section in the original query, which is always echoed in the

(final) reply in the Question Section when the QR bit is set to 1.

* QNAME (effective): A name actually resolved, which is either the name originally queried, or a name received in a CNAME chain response.

* QNAME (final): The name actually resolved, which is either the name actually queried or else the last name in a CNAME chain response.

Note that, because the definition in [RFC2308] is actually for a different concept than what was in [RFC1034], it would have been better if [RFC2308] had used a different name for that concept. In general use today, QNAME almost always means what is defined above as "QNAME (original)".

Referrals: A type of response in which a server, signaling that it is not (completely) authoritative for an answer, provides the querying resolver with an alternative place to send its query. Referrals can be partial.

A referral arises when a server is not performing recursive service while answering a query. It appears in step 3(b) of the algorithm in [RFC1034], Section 4.3.2.

There are two types of referral response. The first is a downward referral (sometimes described as "delegation response"), where the server is authoritative for some portion of the QNAME. The authority section RRset's RDATA contains the name servers specified at the referred-to zone cut. In normal DNS operation, this kind of response is required in order to find names beneath a delegation. The bare use of "referral" means this kind of referral, and many people believe that this is the only legitimate kind of referral in the DNS.

The second is an upward referral (sometimes described as "root referral"), where the server is not authoritative for any portion of the QNAME. When this happens, the referred-to zone in the authority section is usually the root zone (.). In normal DNS operation, this kind of response is not required for resolution or for correctly answering any query. There is no requirement that any server send upward referrals. Some people regard upward referrals as a sign of a misconfiguration or error. Upward referrals always need some sort of qualifier (such as "upward" or "root"), and are never identified by the bare word "referral".

A response that has only a referral contains an empty answer section. It contains the NS RRset for the referred-to zone in the authority section. It may contain RRs that provide addresses in the additional section. The AA bit is clear.

In the case where the query matches an alias, and the server is not authoritative for the target of the alias but it is authoritative for some name above the target of the alias, the resolution algorithm will produce a response that contains both the authoritative answer for the alias, and also a referral. Such a partial answer and referral response has data in the answer section. It has the NS RRset for the referred-to zone in the authority section. It may contain RRs that provide addresses in the additional section. The AA bit is set, because the first name in the answer section matches the QNAME and the server is authoritative for that answer (see [RFC1035], Section 4.1.1).

5. Resource Records

RR: An acronym for resource record. ([RFC1034], Section 3.6.)

RRset: A set of resource records "with the same label, class and type, but with different data". (Definition from [RFC2181], Section 5) Also spelled RRSet in some documents. As a clarification, "same label" in this definition means "same owner name". In addition, [RFC2181] states that "the TTLs of all RRs in an RRSet must be the same".

Note that RRSIG resource records do not match this definition. [RFC4035] says: "An RRset MAY have multiple RRSIG RRs associated with it. Note that as RRSIG RRs are closely tied to the RRsets whose signatures they contain, RRSIG RRs, unlike all other DNS RR types, do not form RRsets. In particular, the TTL values among RRSIG RRs with a common owner name do not follow the RRset rules described in [RFC2181]."

Master file: "Master files are text files that contain RRs in text form. Since the contents of a zone can be expressed in the form of a list of RRs a master file is most often used to define a zone, though it can be used to list a cache’s contents." (Quoted from [RFC1035], Section 5.) Master files are sometimes called "zone files".

Presentation format: The text format used in master files. This format is shown but not formally defined in [RFC1034] and [RFC1035]. The term "presentation format" first appears in [RFC4034].

EDNS: The extension mechanisms for DNS, defined in [RFC6891]. Sometimes called "EDNS0" or "EDNS(0)" to indicate the version number. EDNS allows DNS clients and servers to specify message sizes larger than the original 512 octet limit, to expand the response code space, and to carry additional options that affect the handling of a DNS query.

OPT: A pseudo-RR (sometimes called a "meta-RR") that is used only to contain control information pertaining to the question-and-answer sequence of a specific transaction. (Definition from [RFC6891], Section 6.1.1) It is used by EDNS.

Owner: "The domain name where a RR is found" (Quoted from [RFC1034], Section 3.6). Often appears in the term "owner name".

SOA field names: DNS documents, including the definitions here, often refer to the fields in the RDATA of an SOA resource record by field name. "SOA" stands for "start of a zone of authority". Those fields are defined in Section 3.3.13 of [RFC1035]. The names (in the order they appear in the SOA RDATA) are MNAME, RNAME, SERIAL, REFRESH, RETRY, EXPIRE, and MINIMUM. Note that the meaning of MINIMUM field is updated in Section 4 of [RFC2308]; the new definition is that the MINIMUM field is only "the TTL to be used for negative responses". This document tends to use field names instead of terms that describe the fields.

TTL: The maximum "time to live" of a resource record. "A TTL value is an unsigned number, with a minimum value of 0, and a maximum value of 2147483647. That is, a maximum of 2^31 - 1. When transmitted, the TTL is encoded in the less significant 31 bits of the 32 bit TTL field, with the most significant, or sign, bit set to zero." (Quoted from [RFC2181], Section 8) (Note that [RFC1035] erroneously stated that this is a signed integer; that was fixed by [RFC2181].)

The TTL "specifies the time interval that the resource record may be cached before the source of the information should again be consulted". (Quoted from [RFC1035], Section 3.2.1) Also: "the time interval (in seconds) that the resource record may be cached before it should be discarded". (Quoted from [RFC1035], Section 4.1.3). Despite being defined for a resource record, the TTL of every resource record in an RRset is required to be the same ([RFC2181], Section 5.2).

The reason that the TTL is the maximum time to live is that a cache operator might decide to shorten the time to live for operational purposes, such as if there is a policy to disallow TTL values over a certain number. Some servers are known to ignore

the TTL on some RRsets (such as when the authoritative data has a very short TTL) even though this is against the advice in RFC 1035. An RRset can be flushed from the cache before the end of the TTL interval, at which point the value of the TTL becomes unknown because the RRset with which it was associated no longer exists.

There is also the concept of a "default TTL" for a zone, which can be a configuration parameter in the server software. This is often expressed by a default for the entire server, and a default for a zone using the $TTL directive in a zone file. The $TTL directive was added to the master file format by [RFC2308].

Class independent: A resource record type whose syntax and semantics are the same for every DNS class. A resource record type that is not class independent has different meanings depending on the DNS class of the record, or the meaning is undefined for some class. Most resource record types are defined for class 1 (IN, the Internet), but many are undefined for other classes.

Address records: Records whose type is A or AAAA. [RFC2181] informally defines these as "(A, AAAA, etc)". Note that new types of address records could be defined in the future.

6. DNS Servers and Clients

This section defines the terms used for the systems that act as DNS clients, DNS servers, or both. In the RFCs, DNS servers are sometimes called "name servers", "nameservers", or just "servers". There is no formal definition of DNS server, but the RFCs generally assume that it is an Internet server that listens for queries and sends responses using the DNS protocol defined in [RFC1035] and its successors.

It is important to note that the terms "DNS server" and "name server" require context in order to understand the services being provided. Both authoritative servers and recursive resolvers are often called "DNS servers" and "name servers" even though they serve different roles (but may be part of the same software package).

For terminology specific to the public DNS root server system, see [RSSAC026]. That document defines terms such as "root server", "root server operator", and terms that are specific to the way that the root zone of the public DNS is served.

Resolver: A program "that extract[s] information from name servers in response to client requests." (Quoted from [RFC1034], Section 2.4) A resolver performs queries for a name, type, and

class, and receives responses. The logical function is called "resolution". In practice, the term is usually referring to some specific type of resolver (some of which are defined below), and understanding the use of the term depends on understanding the context.

A related term is "resolve", which is not formally defined in [RFC1034] or [RFC1035]. An imputed definition might be "asking a question that consists of a domain name, class, and type, and receiving some sort of response". Similarly, an imputed definition of "resolution" might be "the response received from resolving".

Stub resolver: A resolver that cannot perform all resolution itself. Stub resolvers generally depend on a recursive resolver to undertake the actual resolution function. Stub resolvers are discussed but never fully defined in Section 5.3.1 of [RFC1034]. They are fully defined in Section 6.1.3.1 of [RFC1123].

Iterative mode: A resolution mode of a server that receives DNS queries and responds with a referral to another server. Section 2.3 of [RFC1034] describes this as "The server refers the client to another server and lets the client pursue the query". A resolver that works in iterative mode is sometimes called an "iterative resolver". See also "iterative resolution" later in this section.

Recursive mode: A resolution mode of a server that receives DNS queries and either responds to those queries from a local cache or sends queries to other servers in order to get the final answers to the original queries. Section 2.3 of [RFC1034] describes this as "The first server pursues the query for the client at another server". Section 4.3.1 of [RFC1034] says "in [recursive] mode the name server acts in the role of a resolver and returns either an error or the answer, but never referrals." That same section also says "The recursive mode occurs when a query with RD set arrives at a server which is willing to provide recursive service; the client can verify that recursive mode was used by checking that both RA and RD are set in the reply."

A server operating in recursive mode may be thought of as having a name server side (which is what answers the query) and a resolver side (which performs the resolution function). Systems operating in this mode are commonly called "recursive servers". Sometimes they are called "recursive resolvers". In practice it is not possible to know in advance whether the server that one is querying will also perform recursion; both terms can be observed in use interchangeably.

Recursive resolver: A resolver that acts in recursive mode. In general, a recursive resolver is expected to cache the answers it receives (which would make it a full-service resolver), but some recursive resolvers might not cache.

[RFC4697] tried to differentiate between a recursive resolver and an iterative resolver.

Recursive query: A query with the Recursion Desired (RD) bit set to 1 in the header. (See Section 4.1.1 of [RFC1035].) If recursive service is available and is requested by the RD bit in the query, the server uses its resolver to answer the query. (See Section 4.3.2 of [RFC1035].)

Non-recursive query: A query with the Recursion Desired (RD) bit set to 0 in the header. A server can answer non-recursive queries using only local information: the response contains either an error, the answer, or a referral to some other server "closer" to the answer. (See Section 4.3.1 of [RFC1035].)

Iterative resolution: A name server may be presented with a query that can only be answered by some other server. The two general approaches to dealing with this problem are "recursive", in which the first server pursues the query on behalf of the client at another server, and "iterative", in which the server refers the client to another server and lets the client pursue the query there. (See Section 2.3 of [RFC1034].)

In iterative resolution, the client repeatedly makes non-recursive queries and follows referrals and/or aliases. The iterative resolution algorithm is described in Section 5.3.3 of [RFC1034].

Full resolver: This term is used in [RFC1035], but it is not defined there. RFC 1123 defines a "full-service resolver" that may or may not be what was intended by "full resolver" in [RFC1035]. This term is not properly defined in any RFC.

Full-service resolver: Section 6.1.3.1 of [RFC1123] defines this term to mean a resolver that acts in recursive mode with a cache (and meets other requirements).

Priming: "The act of finding the list of root servers from a configuration that lists some or all of the purported IP addresses of some or all of those root servers." (Quoted from [RFC8109], Section 2.) In order to operate in recursive mode, a resolver needs to know the address of at least one root server. Priming is most often done from a configuration setting that contains a list of authoritative servers for the root zone.

Root hints: "Operators who manage a DNS recursive resolver typically need to configure a 'root hints file'. This file contains the names and IP addresses of the authoritative name servers for the root zone, so the software can bootstrap the DNS resolution process. For many pieces of software, this list comes built into the software." (Quoted from [IANA_RootFiles]) This file is often used in priming.

Negative caching: "The storage of knowledge that something does not exist, cannot give an answer, or does not give an answer." (Quoted from [RFC2308], Section 1)

Authoritative server: "A server that knows the content of a DNS zone from local knowledge, and thus can answer queries about that zone without needing to query other servers." (Quoted from [RFC2182], Section 2.) An authoritative server is named in the NS ("name server") record in a zone. It is a system that responds to DNS queries with information about zones for which it has been configured to answer with the AA flag in the response header set to 1. It is a server that has authority over one or more DNS zones. Note that it is possible for an authoritative server to respond to a query without the parent zone delegating authority to that server. Authoritative servers also provide "referrals", usually to child zones delegated from them; these referrals have the AA bit set to 0 and come with referral data in the Authority and (if needed) the Additional sections.

Authoritative-only server: A name server that only serves authoritative data and ignores requests for recursion. It will "not normally generate any queries of its own. Instead, it answers non-recursive queries from iterative resolvers looking for information in zones it serves." (Quoted from [RFC4697], Section 2.4) In this case, "ignores requests for recursion" means "responds to requests for recursion with responses indicating that recursion was not performed".

Zone transfer: The act of a client requesting a copy of a zone and an authoritative server sending the needed information. (See Section 7 for a description of zones.) There are two common standard ways to do zone transfers: the AXFR ("Authoritative Transfer") mechanism to copy the full zone (described in [RFC5936], and the IXFR ("Incremental Transfer") mechanism to copy only parts of the zone that have changed (described in [RFC1995]). Many systems use non-standard methods for zone transfer outside the DNS protocol.

Slave server: See "Secondary server".

Secondary server: "An authoritative server which uses zone transfer to retrieve the zone" (Quoted from [RFC1996], Section 2.1). Secondary servers are also discussed in [RFC1034]. [RFC2182] describes secondary servers in more detail. Although early DNS RFCs such as [RFC1996] referred to this as a "slave", the current common usage has shifted to calling it a "secondary".

Master server: See "Primary server".

Primary server: "Any authoritative server configured to be the source of zone transfer for one or more [secondary] servers" (Quoted from [RFC1996], Section 2.1) or, more specifically, "an authoritative server configured to be the source of AXFR or IXFR data for one or more [secondary] servers" (Quoted from [RFC2136]). Primary servers are also discussed in [RFC1034]. Although early DNS RFCs such as [RFC1996] referred to this as a "master", the current common usage has shifted to "primary".

Primary master: "The primary master is named in the zone's SOA MNAME field and optionally by an NS RR". (Quoted from [RFC1996], Section 2.1). [RFC2136] defines "primary master" as "Master server at the root of the AXFR/IXFR dependency graph. The primary master is named in the zone's SOA MNAME field and optionally by an NS RR. There is by definition only one primary master server per zone."

The idea of a primary master is only used in [RFC1996] and [RFC2136]. A modern interpretation of the term "primary master" is a server that is both authoritative for a zone and that gets its updates to the zone from configuration (such as a master file) or from UPDATE transactions.

Stealth server: This is "like a slave server except not listed in an NS RR for the zone." (Quoted from [RFC1996], Section 2.1)

Hidden master: A stealth server that is a primary server for zone transfers. "In this arrangement, the master name server that processes the updates is unavailable to general hosts on the Internet; it is not listed in the NS RRset." (Quoted from [RFC6781], Section 3.4.3). An earlier RFC, [RFC4641], said that the hidden master's name "appears in the SOA RRs MNAME field", although in some setups, the name does not appear at all in the public DNS. A hidden master can also be a secondary server for the zone itself.

Forwarding: The process of one server sending a DNS query with the RD bit set to 1 to another server to resolve that query.

Forwarding is a function of a DNS resolver; it is different than simply blindly relaying queries.

[RFC5625] does not give a specific definition for forwarding, but describes in detail what features a system that forwards needs to support. Systems that forward are sometimes called "DNS proxies", but that term has not yet been defined (even in [RFC5625]).

Forwarder: Section 1 of [RFC2308] describes a forwarder as "a nameserver used to resolve queries instead of directly using the authoritative nameserver chain". [RFC2308] further says "The forwarder typically either has better access to the internet, or maintains a bigger cache which may be shared amongst many resolvers." That definition appears to suggest that forwarders normally only query authoritative servers. In current use, however, forwarders often stand between stub resolvers and recursive servers. [RFC2308] is silent on whether a forwarder is iterative-only or can be a full-service resolver.

Policy-implementing resolver: A resolver acting in recursive mode that changes some of the answers that it returns based on policy criteria, such as to prevent access to malware sites or objectionable content. In general, a stub resolver has no idea whether upstream resolvers implement such policy or, if they do, the exact policy about what changes will be made. In some cases, the user of the stub resolver has selected the policy-implementing resolver with the explicit intention of using it to implement the policies. In other cases, policies are imposed without the user of the stub resolver being informed.

Open resolver: A full-service resolver that accepts and processes queries from any (or nearly any) client. This is sometimes also called a "public resolver", although the term "public resolver" is used more with open resolvers that are meant to be open, as compared to the vast majority of open resolvers that are probably misconfigured to be open. Open resolvers are discussed in [RFC5358]

Split DNS: The terms "split DNS" and "split-horizon DNS" have long been used in the DNS community without formal definition. In general, they refer to situations in which DNS servers that are authoritative for a particular set of domains provide partly or completely different answers in those domains depending on the source of the query. The effect of this is that a domain name that is notionally globally unique nevertheless has different meanings for different network users. This can sometimes be the result of a "view" configuration, described below.

[RFC2775], Section 3.8 gives a related definition that is too specific to be generally useful.

View: A configuration for a DNS server that allows it to provide different responses depending on attributes of the query, such as for "split DNS". Typically, views differ by the source IP address of a query, but can also be based on the destination IP address, the type of query (such as AXFR), whether it is recursive, and so on. Views are often used to provide more names or different addresses to queries from "inside" a protected network than to those "outside" that network. Views are not a standardized part of the DNS, but they are widely implemented in server software.

Passive DNS: A mechanism to collect DNS data by storing DNS responses from name servers. Some of these systems also collect the DNS queries associated with the responses, although doing so raises some privacy concerns. Passive DNS databases can be used to answer historical questions about DNS zones such as which values were present at a given time in the past, or when a name was spotted first. Passive DNS databases allow searching of the stored records on keys other than just the name and type, such as "find all names which have A records of a particular value".

Anycast: "The practice of making a particular service address available in multiple, discrete, autonomous locations, such that datagrams sent are routed to one of several available locations." (Quoted from [RFC4786], Section 2) See [RFC4786] for more detail on Anycast and other terms that are specific to its use.

Instance: "When anycast routing is used to allow more than one server to have the same IP address, each one of those servers is commonly referred to as an 'instance'." "An instance of a server, such as a root server, is often referred to as an 'Anycast instance'." (Quoted from [RSSAC026])

Privacy-enabling DNS server: "A DNS server that implements DNS over TLS [RFC7858] and may optionally implement DNS over DTLS [RFC8094]." (Quoted from [RFC8310], Section 2) Other types of DNS servers might also be considerd privacy-enabling, such as those running DNS over HTTPS [I-D.ietf-doh-dns-over-https].

7. Zones

This section defines terms that are used when discussing zones that are being served or retrieved.

Zone: "Authoritative information is organized into units called 'zones', and these zones can be automatically distributed to the

name servers which provide redundant service for the data in a zone." (Quoted from [RFC1034], Section 2.4)

Child: "The entity on record that has the delegation of the domain from the Parent." (Quoted from [RFC7344], Section 1.1)

Parent: "The domain in which the Child is registered." (Quoted from [RFC7344], Section 1.1) Earlier, "parent name server" was defined in [RFC0882] as "the name server that has authority over the place in the domain name space that will hold the new domain". (Note that [RFC0882] was obsoleted by [RFC1034] and [RFC1035].) [RFC0819] also has some description of the relationship between parents and children.

Origin:

There are two different uses for this term:

(a) "The domain name that appears at the top of a zone (just below the cut that separates the zone from its parent). The name of the zone is the same as the name of the domain at the zone's origin." (Quoted from [RFC2181], Section 6.) These days, this sense of "origin" and "apex" (defined below) are often used interchangeably.

(b) The domain name within which a given relative domain name appears in zone files. Generally seen in the context of "$ORIGIN", which is a control entry defined in [RFC1035], Section 5.1, as part of the master file format. For example, if the $ORIGIN is set to "example.org.", then a master file line for "www" is in fact an entry for "www.example.org.".

Apex: The point in the tree at an owner of an SOA and corresponding authoritative NS RRset. This is also called the "zone apex". [RFC4033] defines it as "the name at the child's side of a zone cut". The "apex" can usefully be thought of as a data-theoretic description of a tree structure, and "origin" is the name of the same concept when it is implemented in zone files. The distinction is not always maintained in use, however, and one can find uses that conflict subtly with this definition. [RFC1034] uses the term "top node of the zone" as a synonym of "apex", but that term is not widely used. These days, the first sense of "origin" (above) and "apex" are often used interchangeably.

Zone cut: The delimitation point between two zones where the origin of one of the zones is the child of the other zone.

"Zones are delimited by 'zone cuts'. Each zone cut separates a 'child' zone (below the cut) from a 'parent' zone (above the cut)." (Quoted from [RFC2181], Section 6; note that this is barely an ostensive definition.) Section 4.2 of [RFC1034] uses "cuts" instead of "zone cut".

Delegation: The process by which a separate zone is created in the name space beneath the apex of a given domain. Delegation happens when an NS RRset is added in the parent zone for the child origin. Delegation inherently happens at a zone cut. The term is also commonly a noun: the new zone that is created by the act of delegating.

Authoritative data: "All of the RRs attached to all of the nodes from the top node of the zone down to leaf nodes or nodes above cuts around the bottom edge of the zone." (Quoted from [RFC1034], Section 4.2.1) Note that this definition might inadvertently also cause any NS records that appear in the zone to be included, even those that might not truly be authoritative because there are identical NS RRs below the zone cut. This reveals the ambiguity in the notion of authoritative data, because the parent-side NS records authoritatively indicate the delegation, even though they are not themselves authoritative data.

[RFC4033], Section 2, defines "Authoritative RRset" which is related to authoritative data but has a more precise definition.

Lame delegation: "A lame delegations exists when a nameserver is delegated responsibility for providing nameservice for a zone (via NS records) but is not performing nameservice for that zone (usually because it is not set up as a primary or secondary for the zone)." (Quoted from [RFC1912], Section 2.8)

Another definition is that a lame delegation "happens when a name server is listed in the NS records for some domain and in fact it is not a server for that domain. Queries are thus sent to the wrong servers, who don't know nothing (at least not as expected) about the queried domain. Furthermore, sometimes these hosts (if they exist!) don't even run name servers." (Quoted from [RFC1713], Section 2.3)

Glue records: "[Resource records] which are not part of the authoritative data [of the zone], and are address resource records for the [name servers in subzones]. These RRs are only necessary if the name server's name is 'below' the cut, and are only used as part of a referral response." Without glue "we could be faced with the situation where the NS RRs tell us that in order to learn a name server's address, we should contact the server using the

address we wish to learn." (Definition from [RFC1034], Section 4.2.1)

A later definition is that glue "includes any record in a zone file that is not properly part of that zone, including nameserver records of delegated sub-zones (NS records), address records that accompany those NS records (A, AAAA, etc), and any other stray data that might appear" (Quoted from [RFC2181], Section 5.4.1). Although glue is sometimes used today with this wider definition in mind, the context surrounding the [RFC2181] definition suggests it is intended to apply to the use of glue within the document itself and not necessarily beyond.

Bailiwick: "In-bailiwick" is an adjective to describe a name server whose name is either a subdomain of or (rarely) the same as the origin of the zone that contains the delegation to the name server. In-bailiwick name servers may have glue records in their parent zone (using the first of the definitions of "glue records" in the definition above). (The term "bailiwick" means the district or territory where a bailiff or policeman has jurisdiction.)

"In-bailiwick" names are divided into two type of name server names: "in-domain" names and "sibling domain" names.

* In-domain: an adjective to describe a name server whose name is either subordinate to or (rarely) the same as the owner name of the NS resource records. An in-domain name server name MUST have glue records or name resolution fails. For example, a delegation for "child.example.com" may have "in-domain" name server name "ns.child.example.com".

* Sibling domain: a name server’s name that is either subordinate to or (rarely) the same as the zone origin and not subordinate to or the same as the owner name of the NS resource records. Glue records for sibling domains are allowed, but not necessary. For example, a delegation for "child.example.com" in "example.com" zone may have "sibling" name server name "ns.another.example.com".

"Out-of-bailiwick" is the antonym of in-bailiwick. An adjective to describe a name server whose name is not subordinate to or the same as the zone origin. Glue records for out-of-bailiwick name servers are useless. Following table shows examples of delegation types.

Delegation	Parent	Name Server Name	Type
com	.	a.gtld-servers.net	in-bailiwick / sibling domain
net	.	a.gtld-servers.net	in-bailiwick / in-domain
example.org	org	ns.example.org	in-bailiwick / in-domain
example.org	org	ns.ietf.org	in-bailiwick / sibling domain
example.org	org	ns.example.com	out-of-bailiwick
example.jp	jp	ns.example.jp	in-bailiwick / in-domain
example.jp	jp	ns.example.ne.jp	in-bailiwick / sibling domain
example.jp	jp	ns.example.com	out-of-bailiwick

Root zone: The zone of a DNS-based tree whose apex is the zero-length label. Also sometimes called "the DNS root".

Empty non-terminals (ENT): "Domain names that own no resource records but have subdomains that do." (Quoted from [RFC4592], Section 2.2.2.) A typical example is in SRV records: in the name "_sip._tcp.example.com", it is likely that "_tcp.example.com" has no RRsets, but that "_sip._tcp.example.com" has (at least) an SRV RRset.

Delegation-centric zone: A zone that consists mostly of delegations to child zones. This term is used in contrast to a zone that might have some delegations to child zones, but also has many data resource records for the zone itself and/or for child zones. The term is used in [RFC4956] and [RFC5155], but is not defined there.

Occluded name: "The addition of a delegation point via dynamic update will render all subordinate domain names to be in a limbo, still part of the zone, but not available to the lookup process. The addition of a DNAME resource record has the same impact. The subordinate names are said to be ’occluded’." (Quoted from [RFC5936], Section 3.5)

Fast flux DNS: This "occurs when a domain is found in DNS using A records to multiple IP addresses, each of which has a very short Time-to-Live (TTL) value associated with it. This means that the domain resolves to varying IP addresses over a short period of time." (Quoted from [RFC6561], Section 1.1.5, with typo corrected) In addition to having legitimate uses, fast flux DNS can used to deliver malware. Because the addresses change so rapidly, it is difficult to ascertain all the hosts. It should be noted that the technique also works with AAAA records, but such use is not frequently observed on the Internet as of this writing.

Reverse DNS, reverse lookup: "The process of mapping an address to a name is generally known as a ’reverse lookup’, and the IN-

ADDR.ARPA and IP6.ARPA zones are said to support the 'reverse DNS'." (Quoted from [RFC5855], Section 1)

Forward lookup: "Hostname-to-address translation". (Quoted from [RFC2133], Section 6)

arpa: Address and Routing Parameter Area Domain: "The 'arpa' domain was originally established as part of the initial deployment of the DNS, to provide a transition mechanism from the Host Tables that were common in the ARPANET, as well as a home for the IPv4 reverse mapping domain. During 2000, the abbreviation was redesignated to 'Address and Routing Parameter Area' in the hope of reducing confusion with the earlier network name." (Quoted from [RFC3172], Section 2.) .arpa is an "infrastructure domain", a domain whose "role is to support the operating infrastructure of the Internet". (Quoted from [RFC3172], Section 2.) See [RFC3172] for more history of this name.

Service name: "Service names are the unique key in the Service Name and Transport Protocol Port Number registry. This unique symbolic name for a service may also be used for other purposes, such as in DNS SRV records." (Quoted from [RFC6335], Section 5.)

8. Wildcards

Wildcard: [RFC1034] defined "wildcard", but in a way that turned out to be confusing to implementers. For an extended discussion of wildcards, including clearer definitions, see [RFC4592]. Special treatment is given to RRs with owner names starting with the label "*". "Such RRs are called 'wildcards'. Wildcard RRs can be thought of as instructions for synthesizing RRs." (Quoted from [RFC1034], Section 4.3.3)

Asterisk label: "The first octet is the normal label type and length for a 1-octet-long label, and the second octet is the ASCII representation for the '*' character. A descriptive name of a label equaling that value is an 'asterisk label'." (Quoted from [RFC4592], Section 2.1.1)

Wildcard domain name: "A 'wildcard domain name' is defined by having its initial (i.e., leftmost or least significant) label be asterisk label." (Quoted from [RFC4592], Section 2.1.1)

Closest encloser: "The longest existing ancestor of a name." (Quoted from [RFC5155], Section 1.3) An earlier definition is "The node in the zone's tree of existing domain names that has the most labels matching the query name (consecutively, counting from the root label downward). Each match is a 'label match' and the order

of the labels is the same." (Quoted from [RFC4592], Section 3.3.1)

Closest provable encloser: "The longest ancestor of a name that can be proven to exist. Note that this is only different from the closest encloser in an Opt-Out zone." (Quoted from [RFC5155], Section 1.3) See Section 10 for more on "opt-out".

Next closer name: "The name one label longer than the closest provable encloser of a name." (Quoted from [RFC5155], Section 1.3)

Source of Synthesis: "The source of synthesis is defined in the context of a query process as that wildcard domain name immediately descending from the closest encloser, provided that this wildcard domain name exists. 'Immediately descending' means that the source of synthesis has a name of the form: <asterisk label>.<closest encloser>." (Quoted from [RFC4592], Section 3.3.1)

9. Registration Model

Registry: The administrative operation of a zone that allows registration of names within that zone. People often use this term to refer only to those organizations that perform registration in large delegation-centric zones (such as TLDs); but formally, whoever decides what data goes into a zone is the registry for that zone. This definition of "registry" is from a DNS point of view; for some zones, the policies that determine what can go in the zone are decided by zones that are superordinate and not the registry operator.

Registrant: An individual or organization on whose behalf a name in a zone is registered by the registry. In many zones, the registry and the registrant may be the same entity, but in TLDs they often are not.

Registrar: A service provider that acts as a go-between for registrants and registries. Not all registrations require a registrar, though it is common to have registrars involved in registrations in TLDs.

EPP: The Extensible Provisioning Protocol (EPP), which is commonly used for communication of registration information between registries and registrars. EPP is defined in [RFC5730].

WHOIS: A protocol specified in [RFC3912], often used for querying registry databases. WHOIS data is frequently used to associate

registration data (such as zone management contacts) with domain names. The term "WHOIS data" is often used as a synonym for the registry database, even though that database may be served by different protocols, particularly RDAP. The WHOIS protocol is also used with IP address registry data.

RDAP: The Registration Data Access Protocol, defined in [RFC7480], [RFC7481], [RFC7482], [RFC7483], [RFC7484], and [RFC7485]. The RDAP protocol and data format are meant as a replacement for WHOIS.

DNS operator: An entity responsible for running DNS servers. For a zone's authoritative servers, the registrant may act as their own DNS operator, or their registrar may do it on their behalf, or they may use a third-party operator. For some zones, the registry function is performed by the DNS operator plus other entities who decide about the allowed contents of the zone.

Public suffix: "A domain that is controlled by a public registry." (Quoted from [RFC6265], Section 5.3) A common definition for this term is a domain under which subdomains can be registered by third parties, and on which HTTP cookies (which are described in detail in [RFC6265]) should not be set. There is no indication in a domain name whether it is a public suffix; that can only be determined by outside means. In fact, both a domain and a subdomain of that domain can be public suffixes.

There is nothing inherent in a domain name to indicate whether it is a public suffix. One resource for identifying public suffixes is the Public Suffix List (PSL) maintained by Mozilla (http://publicsuffix.org/).

For example, at the time this document is published, the "com.au" domain is listed as a public suffix in the PSL. (Note that this example might change in the future.)

Note that the term "public suffix" is controversial in the DNS community for many reasons, and may be significantly changed in the future. One example of the difficulty of calling a domain a public suffix is that designation can change over time as the registration policy for the zone changes, such as was the case with the "uk" TLD in 2014.

Subordinate and Superordinate: These terms are introduced in [RFC3731] for use in the registration model, but not defined there. Instead, they are given in examples. "For example, domain name 'example.com' has a superordinate relationship to host name ns1.example.com'." "For example, host ns1.example1.com is a

subordinate host of domain example1.com, but it is a not a subordinate host of domain example2.com." (Quoted from [RFC3731], Section 1.1.) These terms are strictly ways of referring to the relationship standing of two domains where one is a subdomain of the other.

10. General DNSSEC

Most DNSSEC terms are defined in [RFC4033], [RFC4034], [RFC4035], and [RFC5155]. The terms that have caused confusion in the DNS community are highlighted here.

DNSSEC-aware and DNSSEC-unaware: These two terms, which are used in some RFCs, have not been formally defined. However, Section 2 of [RFC4033] defines many types of resolvers and validators, including "non-validating security-aware stub resolver", "non-validating stub resolver", "security-aware name server", "security-aware recursive name server", "security-aware resolver", "security-aware stub resolver", and "security-oblivious 'anything'". (Note that the term "validating resolver", which is used in some places in DNSSEC-related documents, is also not defined in those RFCs, but is defined below.)

Signed zone: "A zone whose RRsets are signed and that contains properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records." (Quoted from [RFC4033], Section 2.) It has been noted in other contexts that the zone itself is not really signed, but all the relevant RRsets in the zone are signed. Nevertheless, if a zone that should be signed contains any RRsets that are not signed (or opted out), those RRsets will be treated as bogus, so the whole zone needs to be handled in some way.

It should also be noted that, since the publication of [RFC6840], NSEC records are no longer required for signed zones: a signed zone might include NSEC3 records instead. [RFC7129] provides additional background commentary and some context for the NSEC and NSEC3 mechanisms used by DNSSEC to provide authenticated denial-of-existence responses. NSEC and NSEC3 are described below.

Unsigned zone: Section 2 of [RFC4033] defines this as "a zone that is not signed". Section 2 of [RFC4035] defines this as "A zone that does not include these records [properly constructed DNSKEY, Resource Record Signature (RRSIG), Next Secure (NSEC), and (optionally) DS records] according to the rules in this section". There is an important note at the end of Section 5.2 of [RFC4035] that defines an additional situation in which a zone is considered unsigned: "If the resolver does not support any of the algorithms

listed in an authenticated DS RRset, then the resolver will not be able to verify the authentication path to the child zone. In this case, the resolver SHOULD treat the child zone as if it were unsigned."

NSEC: "The NSEC record allows a security-aware resolver to authenticate a negative reply for either name or type non-existence with the same mechanisms used to authenticate other DNS replies." (Quoted from [RFC4033], Section 3.2.) In short, an NSEC record provides authenticated denial of existence.

"The NSEC resource record lists two separate things: the next owner name (in the canonical ordering of the zone) that contains authoritative data or a delegation point NS RRset, and the set of RR types present at the NSEC RR's owner name." (Quoted from Section 4 of RFC 4034)

NSEC3: Like the NSEC record, the NSEC3 record also provides authenticated denial of existence; however, NSEC3 records mitigate against zone enumeration and support Opt-Out. NSEC3 resource records require associated NSEC3PARAM resource records. NSEC3 and NSEC3PARAM resource records are defined in [RFC5155].

Note that [RFC6840] says that [RFC5155] "is now considered part of the DNS Security Document Family as described by Section 10 of [RFC4033]". This means that some of the definitions from earlier RFCs that only talk about NSEC records should probably be considered to be talking about both NSEC and NSEC3.

Opt-out: "The Opt-Out Flag indicates whether this NSEC3 RR may cover unsigned delegations." (Quoted from [RFC5155], Section 3.1.2.1.) Opt-out tackles the high costs of securing a delegation to an insecure zone. When using Opt-Out, names that are an insecure delegation (and empty non-terminals that are only derived from insecure delegations) don't require an NSEC3 record or its corresponding RRSIG records. Opt-Out NSEC3 records are not able to prove or deny the existence of the insecure delegations. (Adapted from [RFC7129], Section 5.1)

Insecure delegation: "A signed name containing a delegation (NS RRset), but lacking a DS RRset, signifying a delegation to an unsigned subzone." (Quoted from [RFC4956], Section 2.)

Zone enumeration: "The practice of discovering the full content of a zone via successive queries." (Quoted from [RFC5155], Section 1.3.) This is also sometimes called "zone walking". Zone enumeration is different from zone content guessing where the guesser uses a large dictionary of possible labels and sends

successive queries for them, or matches the contents of NSEC3 records against such a dictionary.

Validation: Validation, in the context of DNSSEC, refers to one of the following:

* Checking the validity of DNSSEC signatures

* Checking the validity of DNS responses, such as those including authenticated denial of existence

* Building an authentication chain from a trust anchor to a DNS response or individual DNS RRsets in a response

The first two definitions above consider only the validity of individual DNSSEC components such as the RRSIG validity or NSEC proof validity. The third definition considers the components of the entire DNSSEC authentication chain, and thus requires "configured knowledge of at least one authenticated DNSKEY or DS RR" (as described in [RFC4035], Section 5).

[RFC4033], Section 2, says that a "Validating Security-Aware Stub Resolver... performs signature validation" and uses a trust anchor "as a starting point for building the authentication chain to a signed DNS response", and thus uses the first and third definitions above. The process of validating an RRSIG resource record is described in [RFC4035], Section 5.3.

[RFC5155] refers to validating responses throughout the document, in the context of hashed authenticated denial of existence; this uses the second definition above.

The term "authentication" is used interchangeably with "validation", in the sense of the third definition above. [RFC4033], Section 2, describes the chain linking trust anchor to DNS data as the "authentication chain". A response is considered to be authentic if "all RRsets in the Answer and Authority sections of the response [are considered] to be authentic" (Quoted from [RFC4035]). DNS data or responses deemed to be authentic or validated have a security status of "secure" ([RFC4035], Section 4.3; [RFC4033], Section 5). "Authenticating both DNS keys and data is a matter of local policy, which may extend or even override the [DNSSEC] protocol extensions" (Quoted from [RFC4033], Section 3.1).

The term "verification", when used, is usually synonym for "validation".

Validating resolver: A security-aware recursive name server, security-aware resolver, or security-aware stub resolver that is applying at least one of the definitions of validation (above), as appropriate to the resolution context. For the same reason that the generic term "resolver" is sometimes ambiguous and needs to be evaluated in context (see Section 6), "validating resolver" is a context-sensitive term.

Key signing key (KSK): DNSSEC keys that "only sign the apex DNSKEY RRset in a zone." (Quoted from [RFC6781], Section 3.1)

Zone signing key (ZSK): "DNSSEC keys that can be used to sign all the RRsets in a zone that require signatures, other than the apex DNSKEY RRset." (Quoted from [RFC6781], Section 3.1) Also note that a ZSK is sometimes used to sign the apex DNSKEY RRset.

Combined signing key (CSK): "In cases where the differentiation between the KSK and ZSK is not made, i.e., where keys have the role of both KSK and ZSK, we talk about a Single-Type Signing Scheme." (Quoted from [RFC6781], Section 3.1) This is sometimes called a "combined signing key" or CSK. It is operational practice, not protocol, that determines whether a particular key is a ZSK, a KSK, or a CSK.

Secure Entry Point (SEP): A flag in the DNSKEY RDATA that "can be used to distinguish between keys that are intended to be used as the secure entry point into the zone when building chains of trust, i.e., they are (to be) pointed to by parental DS RRs or configured as a trust anchor. Therefore, it is suggested that the SEP flag be set on keys that are used as KSKs and not on keys that are used as ZSKs, while in those cases where a distinction between a KSK and ZSK is not made (i.e., for a Single-Type Signing Scheme), it is suggested that the SEP flag be set on all keys." (Quoted from [RFC6781], Section 3.2.3.) Note that the SEP flag is only a hint, and its presence or absence may not be used to disqualify a given DNSKEY RR from use as a KSK or ZSK during validation.

The original definition of SEPs was in [RFC3757]. That definition clearly indicated that the SEP was a key, not just a bit in the key. The abstract of [RFC3757] says: "With the Delegation Signer (DS) resource record (RR), the concept of a public key acting as a secure entry point (SEP) has been introduced. During exchanges of public keys with the parent there is a need to differentiate SEP keys from other public keys in the Domain Name System KEY (DNSKEY) resource record set. A flag bit in the DNSKEY RR is defined to indicate that DNSKEY is to be used as a SEP." That definition of

the SEP as a key was made obsolete by [RFC4034], and the definition from [RFC6781] is consistent with [RFC4034].

Trust anchor: "A configured DNSKEY RR or DS RR hash of a DNSKEY RR. A validating security-aware resolver uses this public key or hash as a starting point for building the authentication chain to a signed DNS response. In general, a validating resolver will have to obtain the initial values of its trust anchors via some secure or trusted means outside the DNS protocol." (Quoted from [RFC4033], Section 2)

DNSSEC Policy (DP): A statement that "sets forth the security requirements and standards to be implemented for a DNSSEC-signed zone." (Quoted from [RFC6841], Section 2)

DNSSEC Practice Statement (DPS): "A practices disclosure document that may support and be a supplemental document to the DNSSEC Policy (if such exists), and it states how the management of a given zone implements procedures and controls at a high level." (Quoted from [RFC6841], Section 2)

Hardware security module (HSM): A specialized piece of hardware that is used to create keys for signatures and to sign messages without ever disclosing the private key. In DNSSEC, HSMs are often used to hold the private keys for KSKs and ZSKs and to create the signatures used in RRSIG records at periodic intervals.

Signing software: Authoritative DNS servers that support DNSSEC often contain software that facilitates the creation and maintenance of DNSSEC signatures in zones. There is also stand-alone software that can be used to sign a zone regardless of whether the authoritative server itself supports signing. Sometimes signing software can support particular HSMs as part of the signing process.

11. DNSSEC States

A validating resolver can determine that a response is in one of four states: secure, insecure, bogus, or indeterminate. These states are defined in [RFC4033] and [RFC4035], although the two definitions differ a bit. This document makes no effort to reconcile the two definitions, and takes no position as to whether they need to be reconciled.

Section 5 of [RFC4033] says:

A validating resolver can determine the following 4 states:

Secure: The validating resolver has a trust anchor, has a chain of trust, and is able to verify all the signatures in the response.

Insecure: The validating resolver has a trust anchor, a chain of trust, and, at some delegation point, signed proof of the non-existence of a DS record. This indicates that subsequent branches in the tree are provably insecure. A validating resolver may have a local policy to mark parts of the domain space as insecure.

Bogus: The validating resolver has a trust anchor and a secure delegation indicating that subsidiary data is signed, but the response fails to validate for some reason: missing signatures, expired signatures, signatures with unsupported algorithms, data missing that the relevant NSEC RR says should be present, and so forth.

Indeterminate: There is no trust anchor that would indicate that a specific portion of the tree is secure. This is the default operation mode.

Section 4.3 of [RFC4035] says:

A security-aware resolver must be able to distinguish between four cases:

Secure: An RRset for which the resolver is able to build a chain of signed DNSKEY and DS RRs from a trusted security anchor to the RRset. In this case, the RRset should be signed and is subject to signature validation, as described above.

Insecure: An RRset for which the resolver knows that it has no chain of signed DNSKEY and DS RRs from any trusted starting point to the RRset. This can occur when the target RRset lies in an unsigned zone or in a descendent [sic] of an unsigned zone. In this case, the RRset may or may not be signed, but the resolver will not be able to verify the signature.

Bogus: An RRset for which the resolver believes that it ought to be able to establish a chain of trust but for which it is unable to do so, either due to signatures that for some reason fail to validate or due to missing data that the relevant DNSSEC RRs indicate should be present. This case may indicate an attack but may also indicate a configuration error or some form of data corruption.

Indeterminate: An RRset for which the resolver is not able to determine whether the RRset should be signed, as the resolver is not able to obtain the necessary DNSSEC RRs. This can occur when the security-aware resolver is not able to contact security-aware name servers for the relevant zones.

12. Security Considerations

These definitions do not change any security considerations for the DNS.

13. IANA Considerations

None.

14. References

14.1. Normative References

[IANA_RootFiles]
Internet Assigned Numbers Authority, "IANA Root Files", 2016, <http://www.iana.org/domains/root/files>.

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <https://www.rfc-editor.org/info/rfc882>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1123] Braden, R., Ed., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, DOI 10.17487/RFC1123, October 1989, <https://www.rfc-editor.org/info/rfc1123>.

[RFC1912] Barr, D., "Common DNS Operational and Configuration Errors", RFC 1912, DOI 10.17487/RFC1912, February 1996, <https://www.rfc-editor.org/info/rfc1912>.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, DOI 10.17487/RFC1996, August 1996, <https://www.rfc-editor.org/info/rfc1996>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC2182] Elz, R., Bush, R., Bradner, S., and M. Patton, "Selection and Operation of Secondary DNS Servers", BCP 16, RFC 2182, DOI 10.17487/RFC2182, July 1997, <https://www.rfc-editor.org/info/rfc2182>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC3731] Hollenbeck, S., "Extensible Provisioning Protocol (EPP) Domain Name Mapping", RFC 3731, DOI 10.17487/RFC3731, March 2004, <https://www.rfc-editor.org/info/rfc3731>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC4592] Lewis, E., "The Role of Wildcards in the Domain Name System", RFC 4592, DOI 10.17487/RFC4592, July 2006, <https://www.rfc-editor.org/info/rfc4592>.

[RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <https://www.rfc-editor.org/info/rfc5155>.

[RFC5358] Damas, J. and F. Neves, "Preventing Use of Recursive Nameservers in Reflector Attacks", BCP 140, RFC 5358, DOI 10.17487/RFC5358, October 2008, <https://www.rfc-editor.org/info/rfc5358>.

[RFC5730] Hollenbeck, S., "Extensible Provisioning Protocol (EPP)", STD 69, RFC 5730, DOI 10.17487/RFC5730, August 2009, <https://www.rfc-editor.org/info/rfc5730>.

[RFC5855] Abley, J. and T. Manderson, "Nameservers for IPv4 and IPv6 Reverse Zones", BCP 155, RFC 5855, DOI 10.17487/RFC5855, May 2010, <https://www.rfc-editor.org/info/rfc5855>.

[RFC5936] Lewis, E. and A. Hoenes, Ed., "DNS Zone Transfer Protocol (AXFR)", RFC 5936, DOI 10.17487/RFC5936, June 2010, <https://www.rfc-editor.org/info/rfc5936>.

[RFC6561] Livingood, J., Mody, N., and M. O'Reirdan, "Recommendations for the Remediation of Bots in ISP Networks", RFC 6561, DOI 10.17487/RFC6561, March 2012, <https://www.rfc-editor.org/info/rfc6561>.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <https://www.rfc-editor.org/info/rfc6781>.

[RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <https://www.rfc-editor.org/info/rfc6840>.

[RFC6841] Ljunggren, F., Eklund Lowinder, AM., and T. Okubo, "A Framework for DNSSEC Policies and DNSSEC Practice Statements", RFC 6841, DOI 10.17487/RFC6841, January 2013, <https://www.rfc-editor.org/info/rfc6841>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

[RFC7344] Kumari, W., Gudmundsson, O., and G. Barwood, "Automating DNSSEC Delegation Trust Maintenance", RFC 7344, DOI 10.17487/RFC7344, September 2014, <https://www.rfc-editor.org/info/rfc7344>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <https://www.rfc-editor.org/info/rfc7719>.

[RFC8310] Dickinson, S., Gillmor, D., and T. Reddy, "Usage Profiles for DNS over TLS and DNS over DTLS", RFC 8310, DOI 10.17487/RFC8310, March 2018, <https://www.rfc-editor.org/info/rfc8310>.

14.2. Informative References

[I-D.ietf-doh-dns-over-https]
Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", draft-ietf-doh-dns-over-https-14 (work in progress), August 2018.

[IANA_Resource_Registry]
Internet Assigned Numbers Authority, "Resource Record (RR) TYPEs", 2017, <http://www.iana.org/assignments/dns-parameters/>.

[RFC0819] Su, Z. and J. Postel, "The Domain Naming Convention for Internet User Applications", RFC 819, DOI 10.17487/RFC0819, August 1982, <https://www.rfc-editor.org/info/rfc819>.

[RFC0952] Harrenstien, K., Stahl, M., and E. Feinler, "DoD Internet host table specification", RFC 952, DOI 10.17487/RFC0952, October 1985, <https://www.rfc-editor.org/info/rfc952>.

[RFC1713] Romao, A., "Tools for DNS debugging", FYI 27, RFC 1713, DOI 10.17487/RFC1713, November 1994, <https://www.rfc-editor.org/info/rfc1713>.

[RFC1995] Ohta, M., "Incremental Zone Transfer in DNS", RFC 1995, DOI 10.17487/RFC1995, August 1996, <https://www.rfc-editor.org/info/rfc1995>.

[RFC2133] Gilligan, R., Thomson, S., Bound, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 2133, DOI 10.17487/RFC2133, April 1997, <https://www.rfc-editor.org/info/rfc2133>.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, DOI 10.17487/RFC2775, February 2000, <https://www.rfc-editor.org/info/rfc2775>.

[RFC3172] Huston, G., Ed., "Management Guidelines & Operational Requirements for the Address and Routing Parameter Area Domain ("arpa")", BCP 52, RFC 3172, DOI 10.17487/RFC3172, September 2001, <https://www.rfc-editor.org/info/rfc3172>.

[RFC3425] Lawrence, D., "Obsoleting IQUERY", RFC 3425, DOI 10.17487/RFC3425, November 2002, <https://www.rfc-editor.org/info/rfc3425>.

[RFC3757] Kolkman, O., Schlyter, J., and E. Lewis, "Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag", RFC 3757, DOI 10.17487/RFC3757, April 2004, <https://www.rfc-editor.org/info/rfc3757>.

[RFC3912] Daigle, L., "WHOIS Protocol Specification", RFC 3912, DOI 10.17487/RFC3912, September 2004, <https://www.rfc-editor.org/info/rfc3912>.

[RFC4641] Kolkman, O. and R. Gieben, "DNSSEC Operational Practices", RFC 4641, DOI 10.17487/RFC4641, September 2006, <https://www.rfc-editor.org/info/rfc4641>.

[RFC4697] Larson, M. and P. Barber, "Observed DNS Resolution Misbehavior", BCP 123, RFC 4697, DOI 10.17487/RFC4697, October 2006, <https://www.rfc-editor.org/info/rfc4697>.

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <https://www.rfc-editor.org/info/rfc4786>.

[RFC4956] Arends, R., Kosters, M., and D. Blacka, "DNS Security (DNSSEC) Opt-In", RFC 4956, DOI 10.17487/RFC4956, July 2007, <https://www.rfc-editor.org/info/rfc4956>.

[RFC5625] Bellis, R., "DNS Proxy Implementation Guidelines", BCP 152, RFC 5625, DOI 10.17487/RFC5625, August 2009, <https://www.rfc-editor.org/info/rfc5625>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.

[RFC5892] Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.

[RFC5893] Alvestrand, H., Ed. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <https://www.rfc-editor.org/info/rfc5893>.

[RFC5894] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <https://www.rfc-editor.org/info/rfc5894>.

[RFC6055] Thaler, D., Klensin, J., and S. Cheshire, "IAB Thoughts on Encodings for Internationalized Domain Names", RFC 6055, DOI 10.17487/RFC6055, February 2011, <https://www.rfc-editor.org/info/rfc6055>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, DOI 10.17487/RFC6303, July 2011, <https://www.rfc-editor.org/info/rfc6303>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <https://www.rfc-editor.org/info/rfc6335>.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, DOI 10.17487/RFC6365, September 2011, <https://www.rfc-editor.org/info/rfc6365>.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, DOI 10.17487/RFC6672, June 2012, <https://www.rfc-editor.org/info/rfc6672>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

[RFC7129] Gieben, R. and W. Mekking, "Authenticated Denial of Existence in the DNS", RFC 7129, DOI 10.17487/RFC7129, February 2014, <https://www.rfc-editor.org/info/rfc7129>.

[RFC7480] Newton, A., Ellacott, B., and N. Kong, "HTTP Usage in the Registration Data Access Protocol (RDAP)", RFC 7480, DOI 10.17487/RFC7480, March 2015, <https://www.rfc-editor.org/info/rfc7480>.

[RFC7481] Hollenbeck, S. and N. Kong, "Security Services for the Registration Data Access Protocol (RDAP)", RFC 7481, DOI 10.17487/RFC7481, March 2015, <https://www.rfc-editor.org/info/rfc7481>.

[RFC7482] Newton, A. and S. Hollenbeck, "Registration Data Access Protocol (RDAP) Query Format", RFC 7482, DOI 10.17487/RFC7482, March 2015, <https://www.rfc-editor.org/info/rfc7482>.

[RFC7483] Newton, A. and S. Hollenbeck, "JSON Responses for the Registration Data Access Protocol (RDAP)", RFC 7483, DOI 10.17487/RFC7483, March 2015, <https://www.rfc-editor.org/info/rfc7483>.

[RFC7484] Blanchet, M., "Finding the Authoritative Registration Data (RDAP) Service", RFC 7484, DOI 10.17487/RFC7484, March 2015, <https://www.rfc-editor.org/info/rfc7484>.

[RFC7485] Zhou, L., Kong, N., Shen, S., Sheng, S., and A. Servin, "Inventory and Analysis of WHOIS Registration Objects", RFC 7485, DOI 10.17487/RFC7485, March 2015, <https://www.rfc-editor.org/info/rfc7485>.

[RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <https://www.rfc-editor.org/info/rfc7858>.

[RFC8094] Reddy, T., Wing, D., and P. Patil, "DNS over Datagram Transport Layer Security (DTLS)", RFC 8094, DOI 10.17487/RFC8094, February 2017, <https://www.rfc-editor.org/info/rfc8094>.

[RFC8109] Koch, P., Larson, M., and P. Hoffman, "Initializing a DNS Resolver with Priming Queries", BCP 209, RFC 8109, DOI 10.17487/RFC8109, March 2017, <https://www.rfc-editor.org/info/rfc8109>.

[RSSAC026] Root Server System Advisory Committee (RSSAC), "RSSAC Lexicon", 2017, <https://www.icann.org/en/system/files/files/rssac-026-14mar17-en.pdf>.

Appendix A. Definitions Updated by this Document

The following definitions from RFCs are updated by this document:

- Forwarder in [RFC2308]
- QNAME in [RFC2308]
- Secure Entry Point (SEP) in [RFC3757]; note, however, that this RFC is already obsolete

Appendix B. Definitions First Defined in this Document

The following definitions are first defined in this document:

- "Alias" in Section 2
- "Apex" in Section 7
- "arpa" in Section 7
- "Bailiwick" in Section 7
- "Class independent" in Section 5
- "Delegation-centric zone" in Section 7
- "Delegation" in Section 7
- "DNS operator" in Section 9
- "DNSSEC-aware" in Section 10
- "DNSSEC-unaware" in Section 10
- "Forwarding" in Section 6
- "Full resolver" in Section 6
- "Fully qualified domain name" in Section 2
- "Global DNS" in Section 2
- "Hardware Security Module (HSM)" in Section 10
- "Host name" in Section 2
- "IDN" in Section 2
- "In-bailiwick" in Section 7
- "Iterative resolution" in Section 6
- "Label" in Section 2
- "Locally served DNS zone" in Section 2
- "Naming system" in Section 2

- "Negative response" in Section 3
- "Non-recursive query" in Section 6
- "Open resolver" in Section 6
- "Out-of-bailiwick" in Section 7
- "Passive DNS" in Section 6
- "Policy-implementing resolver" in Section 6
- "Presentation format" in Section 5
- "Priming" in Section 6
- "Private DNS" in Section 2
- "Recursive resolver" in Section 6
- "Referrals" in Section 4
- "Registrant" in Section 9
- "Registrar" in Section 9
- "Registry" in Section 9
- "Root zone" in Section 7
- "Secure Entry Point (SEP)" in Section 10
- "Signing software" in Section 10
- "Split DNS" in Section 6
- "Stub resolver" in Section 6
- "Subordinate" in Section 8
- "Superordinate" in Section 8
- "TLD" in Section 2
- "Validating resolver" in Section 10
- "Validation" in Section 10

- "View" in Section 6
- "Zone transfer" in Section 6

Index

Acknowledgements

The following is the Acknowledgements for RFC 7719.

The authors gratefully acknowledge all of the authors of DNS-related RFCs that proceed this one. Comments from Tony Finch, Stephane Bortzmeyer, Niall O'Reilly, Colm MacCarthaigh, Ray Bellis, John Kristoff, Robert Edmonds, Paul Wouters, Shumon Huque, Paul Ebersman, David Lawrence, Matthijs Mekking, Casey Deccio, Bob Harold, Ed Lewis, John Klensin, David Black, and many others in the DNSOP Working Group helped shape RFC 7719.

Most of the major changes between RFC 7719 and this document came from active discussion on the DNSOP WG. Specific people who contributed material to this document include: Bob Harold, Dick Franks, Evan Hunt, John Dickinson, Mark Andrews, Martin Hoffmann, Paul Vixie, Peter Koch, Duane Wessels, Allison Mankin, Giovane Moura, Roni Even, Dan Romascanu, and Vladmir Cunat.

Authors' Addresses

Paul Hoffman
ICANN

Email: paul.hoffman@icann.org

Andrew Sullivan

Email: ajs@anvilwalrusden.com

Kazunori Fujiwara
Japan Registry Services Co., Ltd.
Chiyoda First Bldg. East 13F, 3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: fujiwara@jprs.co.jp

Internet Engineering Task Force
Internet-Draft
Intended status: Experimental
Expires: September 2, 2018

M. Sivaraman
S. Morris
R. Bellis
W. Krecicki
Internet Systems Consortium

March 1, 2018

DNS Catalog Zones
draft-muks-dnsop-dns-catalog-zones-04

Abstract

This document describes a method for automatic DNS zone provisioning among DNS primary and secondary nameservers by storing and transferring the catalog of zones to be provisioned as one or more regular DNS zones.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 2, 2018.

Table of Contents

1. Introduction

The data in a DNS zone is synchronized amongst its primary and secondary nameservers using AXFR and IXFR. However, the list of zones served by the primary (called a catalog in [RFC1035]) is not

automatically synchronized with the secondaries. To add or remove a zone, the administrator of a DNS nameserver farm not only has to add or remove the zone from the primary, they must also add/remove the zone from all secondaries, either manually or via an external application. This can be both inconvenient and error-prone; it will also be dependent on the nameserver implementation.

This document describes a method in which the catalog is represented as a regular DNS zone (called a "catalog zone" here), and transferred using DNS zone transfers. As zones are added to or removed from the catalog zone, the changes are propagated to the secondary nameservers in the normal way. The secondary nameservers then add/remove/modify the zones they serve in accordance with the changes to the zone.

The contents and representation of catalog zones are described in Section 3. Nameserver behavior is described in Section 6.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Catalog zone: A DNS zone containing a DNS catalog, that is, a list of DNS zones and associated zone configuration.

Member zone: A DNS zone whose configuration is published inside a catalog zone.

Zone property: A configuration parameter of a zone, sometimes also called a zone option, represented as a key/value pair.

$CATZ: Used in examples as a placeholder to represent the domain name of the catalog zone itself (c.f. $ORIGIN).

3. Description

A catalog zone is a specially crafted DNS zone that contains, as DNS zone data:

- A list of DNS zones (called "member zones").
- Default zone configuration information common to all member zones.
- Zone-specific configuration information.

An implementation of catalog zones MAY allow the catalog to contain other catalog zones as member zones, but default zone configuration present in a catalog zone only applies to its immediate member zones.

Although the contents of a catalog zone are interpreted and acted upon by nameservers, a catalog zone is a regular DNS zone and so must adhere to the standards for such zones.

A catalog zone is primarily intended for the management of a farm of authoritative nameservers. It is not expected that the content of catalog zones will be accessible from any recursive nameserver.

4. Catalog Zone Structure

4.1. SOA and NS Records

As with any other DNS zone, a catalog zone MUST have a syntactically correct SOA record and one or more NS records at its apex.

The SOA record's SERIAL, REFRESH, RETRY and EXPIRE fields [RFC1035] are used during zone transfer. A catalog zone's SOA SERIAL field MUST increase when an update is made to the catalog zone's contents as per serial number arithmetic defined in [RFC1982]. Otherwise, secondary nameservers might not notice updates to the catalog zone's contents.

Should the zone be made available for querying, the SOA record's MINIMUM field's value is the negative cache time (as defined in [RFC2308]). Since recursive nameservers are not expected to be able to access (and subsequently cache) entries from a catalog zone a value of zero (0) is RECOMMENDED.

Since there is no requirement to be able to query the catalog zone via recursive namservers the NS records at the apex will not be used and no parent delegation is required. However, they are still required so that catalog zones are syntactically correct DNS zones. Any valid DNS name can be used in the NSDNAME field of such NS records [RFC1035] and they MUST be ignored. A single NS RR with an NSDNAME field containing the absolute name "invalid." is RECOMMENDED [RFC2606].

4.2. Zone Data

A catalog zone contains a set of key/value pairs, where each key is encapsulated within the owner name of a DNS RR and the corresponding value is stored in the RR's RDATA. The specific owner name depends on whether the property relates to the catalog zone itself, a member

zone thereof, or to default zone properties described in Section 4.3. The owner names are case insensitive.

4.2.1. Resource Record Format

Each key/value pair has a defined data type, and each data type accordingly uses a particular RR TYPE to represent its possible values, as specified in Section 5.

The general form of a catalog zone record is as follows:

```
[<unique-id>.]<key>.<path>.$CATZ 0 IN <RRTYPE> <value>
```

where <path> is a sequence of labels with values depending on the purpose (and hence position) of the record within the catalog zone (see Section 4.3) and where the <unique-id> prefix is only present for multi-valued properties (see Section 4.2.2).

NB: Catalog zones use some RR TYPEs (such as PTR) with alternate semantics to those originally defined for them. Although this may be controversial, the situation is similar to other similar zone-based representations such as response-policy zones [RPZ].

The CLASS field of every RR in a catalog zone MUST be IN (1). This is because some RR TYPEs such as APL used by catalog zones are defined only for the IN class.

The TTL field’s value is not specially defined by this memo. Catalog zones are for authoritative nameserver management only and are not intended for general querying via recursive resolvers and therefore a value of zero (0) is RECOMMENDED.

It is an error for any single owner name within a catalog zone (other than the apex of the zone itself) to have more than one RR associated with it.

4.2.2. Multi-valued Properties

Some properties do not represent single values but instead represent a collection of values. The specification for each property describes whether it is single-valued or multi-valued. A multi-valued property is encoded as multiple RRs where the owner name of each individual RR contains a unique (user specified) DNS label.

So, while a single-valued key might be represented like this:

```
<key>.<path>.$CATZ IN TXT "value"
```

a multi-valued key would be represented like this:

```
<unique-id-1>.<key>.<path>.$CATZ IN TXT "value 1"
<unique-id-2>.<key>.<path>.$CATZ IN TXT "value 2"
...
```

NB: a property that is specified to be multi-valued MUST be encoded using the unique prefixed key syntax even if there is only one value present.

The specification of any multi-valued property MUST document whether the collection represents either an ordered or un-ordered list. In the former case the ordering of the prefixes according to the usual DNS canonical name ordering will determine the sort order.

4.2.3. Vendor-specific Properties

TBD: Prepare a list of zone configuration properties that are common to DNS implementations. This is so that a company may manage a catalog zone using a Windows DNS server as the primary, and a secondary nameserver hosting service may pick up the common properties and may use a different nameserver implementation such as BIND or NSD on a POSIX operating system to serve it.

TBD: We may specify that unrecognized zone property names must be ignored, or that nameserver specific properties must be specified using the "x-" prefix similar to MIME type naming.

TBD: Any list of zone properties is ideally maintained as a registry rather than within this memo.

4.3. Zone Structure

4.3.1. List of Member Zones

The list of member zones is specified as a multi-valued collection of domain names under the owner name "zones" where "zones" is a direct child domain of the catalog zone.

The names of member zones are represented on the RDATA side (instead of as a part of owner names) so that all valid domain names may be represented regardless of their length [RFC1035].

For example, if a catalog zone lists three zones "example.com.", "example.net." and "example.org.", the RRs would appear as follows:

```
<m-unique-1>.zones.$CATZ 0 IN PTR example.com.
<m-unique-2>.zones.$CATZ 0 IN PTR example.net.
<m-unique-3>.zones.$CATZ 0 IN PTR example.org.
```

where <m-unique-N> is a label that uniquely tags each record in the collection, as described in Section 4.2.2.

Although any legal label could be used for <m-unique-N> it is RECOMMENDED that it be a value deterministically derived from the fully-qualified member zone name. The BIND9 implementation uses the 40 character hexadecimal representation of the SHA-1 digest [FIPS.180-4.2015] of the lower-cased member zone name as encoded in uncompressed wire format.

4.3.2. Catalog Zone Schema Version

The catalog zone schema version is specified by an unsigned integer property with the property name "version". All catalog zones MUST have this property present. Primary and secondary nameservers MUST NOT use catalog zones with an unexpected value in this property, but they may be transferred as ordinary zones. For this memo, the "version" property value MUST be set to 2, i.e.

```
version.$CATZ 0 IN TXT "2"
```

NB: Version 1 was used in a draft version of this memo and reflected the implementation first found in BIND 9.11.

4.3.3. Default Zone Configuration

Default zone configuration comprises a set of properties that are applied to all member zones listed in the catalog zone unless overridden my member zone-specific information.

All such properties are stored as child nodes of the owner name "defaults" itself a direct child node of the catalog zone, e.g.:

```
example-prop.defaults.$CATZ 0 IN TXT "Example"
```

4.3.4. Zone Properties Specific to a Member Zone

Default zone properties can be overridden on a per-zone basis by specifying the property under the the sub-domain associated with the member zone in the list of zones, e.g.:

```
example-prop.<m-unique>.zones.$CATZ 0 IN TXT "Example"
```

where "m-unique" is the label that uniquely identifies the member zone name as described in Section 4.3.1.

NB: when a zone-specific property is multi-valued the owner name will contain two unique identifiers, the left-most tagging being associated with the individual value (<unique-id-N>) and the other (<m-unique>) associated with the member zone itself, e.g.:

```
$ORIGIN <m-unique>.zones.$CATZ
<unique-id-1>.example-prop 0 IN TXT "Value 1"
<unique-id-2>.example-prop 0 IN TXT "Value 2"
...
```

5. Data Types

This section lists the various data types defined for use within catalog zones.

5.1. String

A key with a string value is represented with a TXT RR [RFC1035], e.g.:

```
example-prop.<m-unique>.zones.$CATZ 0 IN TXT "Example"
```

If the RDATA is split into multiple <character-string> elements the MUST be directly concatenated without any separating character.

5.2. Booleans

A key with a boolean value is represented with a TXT RR containing a single <character-string> with a value of "true" for true condition and "false" for false condition, e.g:

```
example-prop.<m-unique>.zones.$CATZ 0 IN TXT "false"
```

The RDATA is case-insensitive.

5.3. Integers

A key with an integer value is specified using a TXT RR containing a single <character-string>.

A signed integer’s TXT RDATA uses the representation of an unsuffixed "integer constant" as defined in the C programming language standard [ISO.9899.1990] (of the type matching a 64-bit signed integer on that platform), with an optional minus prefix.

An unsigned integer’s TXT RDATA uses the representation of an unsuffixed "integer constant" as defined in the C programming language standard [ISO.9899.1990] (of the type matching a 64-bit unsigned integer on that platform).

For example, a property with an unsigned integer value of 300 would appear as follows:

```
example-prop.<m-unique>.zones.$CATZ 0 IN TXT "300"
```

5.4. Floating-Point Values

A key with a floating-point value is specified using a TXT RR containing a single <character-string>.

A floating-point value’s TXT RDATA uses the representation of an unsuffixed "floating constant" as defined in the C programming language standard [ISO.9899.1990].

For example, a property with an unsigned integer value of 0.15 may appear as follows:

```
example-prop.<m-unique>.zones.$CATZ 0 IN TXT "15e-2"
```

5.5. Domain Name

A key whose value is a domain name is specified using a PTR RR [RFC1035], e.g.:

```
example-prop.defaults.$CATZ 0 IN PTR ns1.example.com.
```

5.6. IP Prefix

A property whose value is an IP network prefix is specified using an APL RR [RFC3123]. The negation flag ("!" in presentation format) may be used to indicate all addresses not included within that prefix, e.g. for use in Access Control Lists, e.g.:

Although a single APL record is capable of containing multiple prefixes, for consistency of representation lists of prefixes MUST use the multi-valued property syntax as documented in Section 4.2.2, e.g.:

```
$ORIGIN <m-unique>.zones.$CATZ
<unique-id-1>.example-prop 0 IN APL ( 1:192.0.2.0/24 )
<unique-id-2>.example-prop 0 IN APL ( !1:0.0.0.0/0 )
```

Implementations MUST accept only the first prefix within each APL record and MUST ignore any subsequent prefixes found therein.

5.7. Single Host Address

A single host address is represented using either an A or AAAA record as appropriate, e.g.:

```
example-prop1.<m-unique>.zones.$CATZ 0 IN A 192.0.2.1
example-prop2.<m-unique>.zones.$CATZ 0 IN AAAA 2001:db8::1
```

6. Nameserver Behavior

6.1. General Requirements

As it is a regular DNS zone, a catalog zone can be transferred using DNS zone transfers among nameservers.

Although they are regular DNS zones, catalog zones contain only information for the management of a set of authoritative nameservers. For this reason, operators may want to limit the systems able to query these zones. It may be inconvenient to serve some contents of catalog zones via DNS queries anyway due to the nature of their representation. A separate method of querying entries inside the catalog zone may be made available by nameserver implementations (see Section 6.3).

Catalog updates should be automatic, i.e., when a nameserver that supports catalog zones completes a zone transfer for a catalog zone, it SHOULD apply changes to the catalog within the running nameserver automatically without any manual intervention.

As with regular zones, primary and secondary nameservers for a catalog zone may be operated by different administrators. The secondary nameservers may be configured to synchronize catalog zones from the primary, but the primary’s administrators may not have any administrative access to the secondaries.

A catalog zone can be updated via DNS UPDATE on a reference primary nameserver, or via zone transfers. Nameservers MAY allow loading and transfer of broken zones with incorrect catalog zone syntax (as they are treated as regular zones), but nameservers MUST NOT process such broken zones as catalog zones. For the purpose of catalog processing, the broken catalogs MUST be ignored. If a broken catalog zone was transferred, the newly transferred catalog zone MUST be ignored (but the older copy of the catalog zone SHOULD be left running subject to values in SOA fields).

If there is a clash between an existing member zone's name and an incoming member zone's name (via transfer or update), the new instance of the zone MUST be ignored and an error SHOULD be logged.

When zones are introduced into a catalog zone, a primary SHOULD first make the new zones available for transfers before making the updated catalog zone available for transfer, or sending NOTIFY for the catalog zone to secondaries. Note that secondary nameservers may attempt to transfer the catalog zone upon refresh timeout, so care must be taken to make the member zones available before any update to the list of member zones is visible in the catalog zone.

When zones are deleted from a catalog zone, a primary MAY delete the member zone immediately after notifying secondaries. It is up to the secondary nameserver to handle this condition correctly.

TBD: Transitive primary-secondary relationships

6.2. Updating Catalog Zones

TBD: Explain updating catalog zones using DNS UPDATE.

6.3. Implementation Notes

Catalog zones on secondary nameservers would have to be setup manually, perhaps as static configuration, similar to how ordinary DNS zones are configured. Members of such catalog zones will be automatically synchronized by the secondary after the catalog zone is configured.

An administrator may want to look at data inside a catalog zone. Typical queries might include dumping the list of member zones, dumping a member zone's effective configuration, querying a specific property value of a member zone, etc. Because of the structure of catalog zones, it may not be possible to perform these queries intuitively, or in some cases, at all, using DNS QUERY. For example it is not possible to enumerate the contents of a multi-valued property (such as the list of member zones) with a single QUERY. Implementations are therefore advised to provide a tool that uses either the output of AXFR or an out-of-band method to perform queries on catalog zones.

7. Security Considerations

As catalog zones are transmitted using DNS zone transfers, it is absolutely essential for these transfers to be protected from unexpected modifications on the route. So, catalog zone transfers SHOULD be authenticated using TSIG [RFC2845]. A primary nameserver

SHOULD NOT serve a catalog zone for transfer without using TSIG and a secondary nameserver SHOULD abandon an update to a catalog zone that was received without using TSIG.

Use of DNS UPDATE [RFC2136] to modify the content of catalog zones SHOULD similarly be authenticated using TSIG.

Zone transfers of member zones SHOULD similarly be authenticated using TSIG [RFC2845]. The TSIG shared secrets used for member zones MUST NOT be mentioned anywhere in the catalog zone data. However, key identifiers may be shared within catalog zones.

Catalog zones do not need to be signed using DNSSEC, their zone transfers being authenticated by TSIG. Signed zones MUST be handled normally by nameservers, and their contents MUST NOT be DNSSEC-validated.

8. IANA Considerations

This document has no IANA actions.

9. Acknowledgements

Catalog zones originated as the chosen method among various proposals that were evaluated at ISC for easy zone management. The chosen method of storing the catalog as a regular DNS zone was proposed by Stephen Morris.

We later discovered that Paul Vixie’s earlier [Metazones] proposal implemented a similar approach and reviewed it. Catalog zones borrows some syntax ideas from Metazones, as both share this scheme of representing the catalog as a regular DNS zone.

Thanks to Brian Conry, Tony Finch, Evan Hunt, Patrik Lundin, Victoria Risk and Carsten Strettman for reviewing draft proposals and offering comments and suggestions.

10. References

10.1. Normative references

[FIPS.180-4.2015]
National Institute of Standards and Technology, "Secure Hash Standard", FIPS PUB 180-4, August 2015, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[ISO.9899.1990]
International Organization for Standardization, "Programming languages - C", ISO Standard 9899, 1990.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <https://www.rfc-editor.org/info/rfc2136>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC2606] Eastlake 3rd, D. and A. Panitz, "Reserved Top Level DNS Names", BCP 32, RFC 2606, DOI 10.17487/RFC2606, June 1999, <https://www.rfc-editor.org/info/rfc2606>.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <https://www.rfc-editor.org/info/rfc2845>.

[RFC3123] Koch, P., "A DNS RR Type for Lists of Address Prefixes (APL RR)", RFC 3123, DOI 10.17487/RFC3123, June 2001, <https://www.rfc-editor.org/info/rfc3123>.

10.2. Informative references

[Metazones]
Vixie, P., "Federated Domain Name Service Using DNS Metazones", 2005, <http://ss.vix.su/~vixie/mz.pdf>.

[RPZ] Vixie, P. and V. Schryver, "DNS Response Policy Zones (DNS RPZ)", 2010, <http://ftp.isc.org/isc/dnsrpz/isc-tn-2010-1.txt>.

Appendix A. Open issues and discussion (to be removed before final publication)

1. Config options

   We want catalog zones to be adopted by multiple DNS implementations. Towards this, we have to generalize zone config options and adopt a minimal set that we can expect most implementations to support.

2. Catalog zone and member zones on different primary nameservers

   Will it be possible to setup a catalog zone on one nameserver as primary, and allow its member zones to be served by different primary nameservers?

3. Transitive relationships

   For a catalog zone, a secondary nameserver may be a primary nameserver to a different set of nameservers in a nameserver farm. In these transitive relationships, zone configuration options (such as also-notify and allow-transfer) may differ based on the location of the primary in the hierarchy. It may not be possible to specify this within a catalog zone.

4. Overriding controls

   A way to override zone config options (as prescribed by the catalog zones) on secondary nameservers was requested. As this would be configured outside catalog zones, it may be better to leave this to implementations.

Appendix B. Change History (to be removed before final publication)

- draft-muks-dnsop-dns-catalog-zones-00
  Initial public draft.

- draft-muks-dnsop-dns-catalog-zones-01
  Added Witold, Ray as authors. Fixed typos, consistency issues. Fixed references. Updated Area. Removed newly introduced custom RR TYPEs. Changed schema version to 1. Changed TSIG requirement from MUST to SHOULD. Removed restrictive language about use of DNS QUERY. When zones are introduced into a catalog zone, a primary SHOULD first make the new zones available for transfers first (instead of MUST). Updated examples, esp. use IPv6 in examples per Fred Baker. Add catalog zone example.

- draft-muks-dnsop-dns-catalog-zones-02

Addressed some review comments by Patrik Lundin.

- draft-muks-dnsop-dns-catalog-zones-03
  Revision bump.

- draft-muks-dnsop-dns-catalog-zones-04
  Reordering of sections into more logical order.
  Separation of multi-valued properties into their own category.

Authors' Addresses

Mukund Sivaraman
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: muks@mukund.org
URI: http://www.isc.org/

Stephen Morris
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: stephen@isc.org
URI: http://www.isc.org/

Ray Bellis
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: ray@isc.org
URI: http://www.isc.org/

Witold Krecicki
Internet Systems Consortium
950 Charter Street
Redwood City, CA 94063
US

Email: wpk@isc.org
URI: http://www.isc.org/

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 20, 2017

T. Lemon
Nominum, Inc.
R. Droms
Cisco, Inc.
W. Kumari
Google

September 16, 2016

Special-Use Names Problem Statement
draft-tldr-sutld-ps-04

Abstract

The Special-Use Domain Names registration policy in RFC 6761 has been shown through experience to present unanticipated challenges. This memo presents a list, intended to be comprehensive, of the problems that have been identified. In addition it reviews the history of Domain Names and summarizes current IETF publications and some publications from other standards organizations relating to special-use domain names.

[Ed note: John Levine suggested we use 'Domain Names' instead of 'Internet Names'; this is the only change between -03 and -04. Please let us know which term you prefer.]

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 20, 2017.

Table of Contents

1. Introduction

One of the key services required to use the Internet is name resolution. Name resolution is the process of translating a human-readable symbolic name into some object or set of objects to which the name refers, most typically one or more IP addresses. These names are often referred to as domain names. When reading this document, care must be taken to not assume that the term Domain Name implies the particular protocol for resolving these names, the Domain Name System [RFC1034]. An excellent presentation on this topic can be found in Domain Names [I-D.lewis-domain-names].

At the time of this writing, the IETF has recently been asked to allocate several new special-use top-level Domain Names. In evaluating the process for additional special-use top-level Domain Names as documented in Special-Use Domain Names [RFC6761], the IETF encountered several different sorts of issues. Because of this, the IETF has decided to investigate the problem and decide if and how the RFC 6761 process can be improved, or whether it should be deprecated.

This document presents a list, believed to be complete, of the problems associated with the allocation of special-use names. In support of the particular set of problems described here, the document also includes documentation of existing practice as it relates to the use of Domain Names, as well as a brief history of domain names, and finally to describe the set of problems that exist as reported by various IETF participants with experience in the various aspects of the problem.

2. Terminology

For the sake of brevity this document uses a number of abbreviations. These are expanded here:

Domain Name A name which serves as input to a host's ordinary name resolution process, for example 'EXAMPLE.ORG'.

SUDN Special Use Domain Name

SUTLDN Special-Use Top-Level Domain Name

IANA Internet Assigned Numbers Authority

ICANN Internet Corporation for Assigned Names and Numbers

3. Problems associated with Special-Use Internet Names

This section presents a list of problems that have been identified with respect to the allocation of special-use names. Solutions to these problems are out of scope. Because of that, problems with solutions to these problems are also out of scope, and will be covered in a separate document.

No assertion is made that any of these problems is more or less important than any other. The point of this is simply to enumerate and briefly describe the problems that have been raised during discussions of the special-use name problem. The degree of detail is intended to be sufficient that that participants in the discussion can agree that the problems they've raised have been adequately described, and no more.

In addition, no assertion is made that all of these problems must be addressed; while each problem has one or more solutions, the solutions may in some cases be mutually contradictory, or may come with costs that do not justify the benefit that would be obtained from solving them.

This is the list of problems:

- IETF and ICANN independently have remit to assign names out of the namespace that Domain Names represent; a formal coordination process does not exist.
- Although IETF and ICANN nominally have authority over this namespace, neither organization can enforce that authority over any third party who wants to just start using a subset of the namespace. Reasons for doing this may include:
  - Unaware that a process exists for allocating such names
  - Intended use is covered by gTLD allocation, but no gTLD allocation is ongoing
  - Intended use is covered by gTLD allocation, don’t want to pay fee
  - Intended use is covered by some IETF process, but don’t want to follow process
  - Intended use is covered by ICANN or IETF process, but expected outcome is refusal
- There is demand for more than one name resolution protocol for Domain Names, but Domain Names contain no metadata to indicate which protocol to use to resolve them.
- When a top-level name is used as a means either of marking the rest of a Domain Name for resolution using a protocol other than DNS, or is used for resolution of names with no global meaning, not all software that processes such names will understand the names’ special meanings. Consequently, any such use results in queries for those names being sent to authoritative servers.
- The RFC6761 process is sufficiently uncertain that some protocol developers have assumed they could not get a name assigned; the process of assigning the first new name following RFC 6761 took more than ten years from beginning to end: longer by a factor of ten than any other part of the protocol development process. Other uses of the process have proceeded more smoothly, but there

is a reasonably justified perception that using this process is likely to be slow and difficult, with an uncertain outcome.

- There is strong resistance within the IETF to assigning names to things outside of the DNS, for a variety of reasons:
  - Requires a mechanism for identifying which of a set of resolution processes is required in order to resolve a particular name.
  - Assertion of authority: there is a sense that the namespace is "owned" by the IETF or by ICANN, and so, if a name is allocated outside of that process, the person or entity that allocated that name should suffer some consequence that would, presumably, deter future circumvention of the official process.
  - More than one name resolution protocol is bad, in the sense that a single protocol is less complicated to implement and deploy.
  - The semantics of alternative resolution protocols may differ from the DNS protocol; DNS has the concept of RRtypes; other protocols may not support RRtypes, or may support some entirely different data structuring mechanism.
  - If there is an IETF process through which a name can be allocated at zero cost other than time, this process will be used as an alternative to allocating the name through ICANN.
  - Some names that might be allocated would be sufficiently generic that other legitimate uses of those names would overlap with a proposed use, so that assigning the name would preclude some future, better use of it.
  - If the IETF allocates a name that some third party or parties believes belongs to them in some way, the IETF could become embroiled in an expensive dispute with those parties.
- In cases where the IETF has made assignments through the RFC 6761 process, technical mistakes have been made due either to insufficiently well-defined process or to a failure to follow the process that was defined in RFC 6761.
- In principal, the RFC 6761 process could be used to document the existence of domain names that are not safe to allocate, and provide information on how those names are used in practice. However, attempts to use RFC6761 to accomplish this [I-D.chapin-additional-reserved-tlds] have not been successful.

- o No process exists for checking documents to make sure they don't accidentally assign names (e.g. RFC 7788).

- o Use of the registry is inconsistent--some SUTLIN RFCs specify registry entries; some don't; some specify delegation, some don't.

- o There exists no safe, non-process-violating mechanism for ad-hoc allocation of special-use names.

- o RFC 6761 uses the term "Domain Name" to describe the thing for which special uses are registered. This creates a great deal of confusion because some readers take "Domain Name" to imply the use of the DNS protocol.

The problems we have stated here represent the current understanding of the authors of the document as to the complete set of problems that have been identified during discussion by the working group on this topic. The remainder of this document provides additional context that will be needed for reasoning about these problems.

4. Existing Practice Regarding SUINs

There are three primary and numerous secondary documents to consider when thinking about the Special-Use Domain Names process.

4.1. Primary SUIN Documents

The primary documents are considered primary because they directly address the IETF's past thoughts on this topic in a general way, and also because they describe what the IETF does in practice. Only one of these documents is an IETF consensus document.

4.1.1. IAB Technical Comment on the Unique DNS Root

This document [RFC2826] is not an IETF consensus document, and appears to have been written to address a different problem than the SUDN problem. However, it speaks directly to several of the key issues that must be considered, and of course coming as it does from the IAB, it is rightly given with a great deal of authority despite not being an IETF consensus document.

This document should be considered required reading for IETF participants who wish to express an informed opinion on the topic of SUDNs. The main points that appear relevant to the specal use names problem are:

- o The Internet requires a globally unique namespace

- Private networks may operate private namespaces, but still require that names in the public namespace be globally unique.

- The Domain Name System [RFC1035] is not the only protocol that may be used for resolving domain names.

- Users cannot be assumed to know how to distinguish between symbols that have local meaning and symbols that have global meaning. Users may therefore share symbols that incorporate domain names with no global meaning (for example, a URL of 'http://mysite.example.corp', where 'example.corp' is a domain allocated privately and informally as described in [SDO-ICANN-COLL]). Such symbols might refer to the object the user intends to share within that user's context, but either refer to some other object any recipient's context, or might not refer to any object at all in a recipient's context. The effect of this is that the user's intended communication will not be able to be understood by the recipients of the communication. This same problem can also occur simply because a single user copies a name from one context in which it has one meaning, into a different context in which it has a different meaning-- for example copying a '.onion' Domain Name out of a TOR browser, where it has meaning, and pasting this name into an ssh client that doesn't support connecting over TOR. [TODO: Consider "labels" instead of "symbols".]

To boil this down even further, we can take the following advice from this document:

- Domain names with unambiguous global meaning are preferable to domain names with local meaning which will be ambiguous. Nevertheless both globally-meaningful and locally-special names are in use and must be supported.

- At the time of the writing of this document the IAB was of the opinion that there might well be more than one name resolution protocol used to resolve domain names.

4.1.2. Special-Use Domain Names

The second important document is Special-Use Domain Names [RFC6761]. RFC6761 represents the current IETF consensus on designating and recording SUDNs. The IETF has experienced problems with the designation process described in RFC6761; these concerns motivate this document. Again, familiarity with RFC6761 is a prerequisite for having an informed opinion on the topic of SUDNs.

RFC 6761 defines two aspects of SUDNs: designating a domain name to have a special purpose and registering that special use in the Special-Use Domain Names registry. The designation process is defined in a single sentence (RFC6761, section 4):

> If it is determined that special handling of a name is required in order to implement some desired new functionality, then an IETF "Standards Action" or "IESG Approval" specification [RFC5226] MUST be published describing the new functionality.

This sentence implies that any designation of a special-use name is subject to the same open review and consensus process as used to produce and publish all other IETF specifications.

The registration process is a purely mechanical process, in which the existence of the newly designated special use name is recorded, with a pointer to a section in the relevant specification document that defines the ways in which special handling is to be applied to the name.

RFC6761 provided the process whereby Multicast DNS [RFC6762] designated ".local" as a special-use name and included it in the Special-Use Names registry. It itself also enumerated a set of names that had been previously used or defined to have special uses prior to the publication of RFC6761. Since there had been no registry for these names prior to the publication of RFC 6761, the documents defining these names could not have added them to the registry.

There are at least several important points to think of with respect to the RFC6761:

- A special-use name may be a name that should be resolved using the DNS protocol with no special handling. An example of this is .ARPA.

- A special-use name may be a name that is resolved using the DNS protocol, requires no special handling in the stub resolver, but requires special handling in the recursive resolver. An example of this would be "10.in-addr.arpa."

- A special-use name may be name that requires special handling in the stub resolver. An example would be a special-use top-level name like ’.local’ which acts as a signal to indicate that the local stub resolver should use a non-DNS protocol for name resolution.

- The current IETF consensus (from a process perspective, not necessarily from the perspective of what would be consensus if the

> IETF were to attempt to produce a new consensus document) is that these purposes for special-use names are valid. [TODO: "Both" implies that there are only two applications; the above bullet points outline 3...]

The term "stub resolver" in this case does not mean "DNS protocol stub resolver." The stub resolver is the entity within a particular software stack that takes a question about a Domain name and answers it. One way a stub resolver can answer such a question is using the DNS protocol, but it is in the stub resolver, as we are using the term here, that the decision as to whether to use a protocol, and if so which protocol, or whether to use a local database of some sort, is made.

RFC6761 does not limit special-use names to TLDs. However, at present, all special-use names registered in the IANA Special-Use Domain Names registry [SDO-IANA-SUDR] are either intended to be resolved using the DNS protocol, or are top-level domains, or both. That is, at present there exist no special-use names which require special handling by stub resolvers and which are not at the top level of the naming hierarchy.

This does mean, however, that at present, RFC6762 requires the use of a special label, '.LOCAL', to indicate to stub resolvers that mDNS[RFC6762] be used to resolve names under that label.

4.1.3. MoU Concerning the Technical Work of the IANA

There exists a Memorandum of Understanding[RFC2860] between the IETF and ICANN (Internet Corporation for Assigned Names and Numbers) which discusses how names and numbers will be managed through the IANA (Internet Assigned Numbers Authority). This document is important to the discussion of SUDNs because, while it delegates authority for managing the Domain Name System Namespace generally to ICANN, it reserves to the IETF the authority that RFC 6761 formalizes:

> Note that (a) assignments of domain names for technical uses (such as domain names for inverse DNS lookup), (b) assignments of specialised address blocks (such as multicast or anycast blocks), and (c) experimental assignments are not considered to be policy issues, and shall remain subject to the provisions of this Section 4.

The above text is an addendum to the following:

> Two particular assigned spaces present policy issues in addition to the technical considerations specified by the IETF: the

assignment of domain names, and the assignment of IP address blocks. These policy issues are outside the scope of this MOU.

In general, then, the assignment of names in the DNS root zone, and the management of the DNS namespace, is a function that is performed by ICANN. However, the MoU specifically exempts domain names assigned for technical use, and uses the example of 'IN-ADDR.ARPA' and 'IP6.ARPA' to illustrate. Both of these names are in the RFC 6761 registry.

The point here is not to say what the implications of this statement in the MoU are, but rather to call the reader's attention to the existence of this statement.

4.2. Secondary documents relating to the SUTLIN question

In addition to these documents, there are several others with which participants in this discussion should be familiar.

4.2.1. Multicast DNS

Multicast DNS [RFC6762] defines the Multicast DNS protocol, which uses the '.LOCAL' SUTLDN. Section 3 describes the semantics of "multicast DNS names." It is of considerable historical importance to note that the -00 version of this document, an individual submission, was published in July of 2001. This version contains substantially the same text in section 3, and was discussed in the DNSEXT working group at IETF 51 in August of 2001[IETF-PRO-51]. The first version of this document designated '.LOCAL.ARPA' as the special-use name. This idea was strongly opposed by DNSEXT working group participants, and as a result the author eventually switched to using '.LOCAL'.

The history of RFC 6762 is documented in substantial detail in Appendix H; some notable milestones include the initial proposal to replace Appletalk's NBP in July 1997, the chartering of the Zeroconf working group in September 1999, allocation of a multicast address for link-local name discovery in April of 2000. A companion requirements document, eventually published as [RFC6760] was first published in September of 2001.

The point of mentioning these dates is so that discussions involving the time when the '.LOCAL' domain was first deployed, and the context in which it was deployed, may be properly informed.

4.2.2. The .onion Special-Use TLD

The .onion Special Use TLD [RFC7686] is important because it is the most recent IETF action on the topic of SUTLDNs; although it does not set new policy, the mere fact of its publication is worth thinking about.

Two important points to consider about this document are that:

- The IETF gained consensus to publish it
- The situation was somewhat forced, both by the fact of its unilateral allocation by the TOR project without following the RFC 6761 process, and because a deadline had been set by the CA/ Browser Forum [SDO-CABF-INT] after which all .onion PKI certificates would expire and no new certificates would be issued, unless the .onion SUTLDN were to be recognized by the IETF.

4.2.3. Locally Served DNS Zones

Locally Served DNS Zones [RFC6303] describes a particular use case for zones that exist by definition, and that are resolved using the DNS protocol, but that cannot have a global meaning, because the host IP addresses they reference are not unique. This applies to a variety of addresses, including Private IPv4 addresses [RFC1918], Unique Local IPv6 Unicast Addresses [RFC4193] (in which this practice was first described) and IANA-Reserved IPv4 Prefix for Shared Address Space [RFC6598].

This use case is distinct from the use-case for SUTLDNs like ’.local’ and ’.onion’ in that the names are resolved using the DNS protocol. But it shares the problem that such names cannot be assumed either to be unique or to be functional in all contexts for all Internet-connected hosts.

4.2.4. Name Collision in the DNS

Name Collision in the DNS [SDO-ICANN-COLL] is a study commissioned by ICANN that attempts to characterize the potential risk to the Internet of adding global DNS delegations for names that were not previously delegated in the DNS, not reserved under any RFC, but also known to be (.local) or surmised to be (.corp) in significant use for special-use-type reasons (local scope DNS, or other resolution protocols altogether).

4.2.5. Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis

Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis [RFC7050] is an example of a document that successfully used the RFC 6761 process to designate '.ipv4only.arpa' as a special-use name; in this case the process worked smoothly and without controversy.

4.2.6. Additional Reserved Top Level Domains

Additional Reserved Top Level Domains [I-D.chapin-additional-reserved-tlds] is an example of a document that attempted to reserve several TLDs identified by ICANN as particularly at risk for collision as special-use domain names with no documented use. This attempt failed.

Although this document failed to gain consensus to publish, the need it was intended to fill still exists. Unfortunately, although a fair amount is known about the use of these names, no document exists that documents how they are used, and why it would be a problem to allocate them. Additionally, to the extent that the uses being made of these names are valid, no document exists indicating when it might make sense to use them, and when it would not make sense to use them (the simplest version of this document would of course say "never use them." If that were the IETF consensus, that would be a good reason not to bother to publish the document.

4.3. Summary

The assignment of Internet Names is not under the sole control of any one organization. ICANN has authority in many cases, and could be considered in some sense the default. IETF has authority in other cases, but only with respect to protocol development. And neither of these authorities can in any practical sense exclude the practice of ad-hoc allocation of names, which can be done by any entity that has control over one or more name servers or resolvers, in the context of any hosts and services that that entity operates.

5. History

Newcomers to the problem of resolving domain names may be under the mistaken impression that the DNS sprang, as in the Greek legend of Athena, directly from Paul Mockapetris' forehead. This is not the case. At the time of the writing of the IAB technical document, memories would have been fresh of the evolutionary process that led to the DNS' dominance as a protocol for domain name resolution.

In fact, in the early days of the Internet, hostnames were resolved using a text file, HOSTS.TXT, which was maintained by a central

authority, the Network Information Center, and distributed to all hosts on the Internet using the File Transfer Protocol (FTP) [RFC0959]. The inefficiency of this process is cited as a reason for the development of the DNS [RFC0882] [RFC0883] in 1983.

However, the transition from HOSTS.TXT to the DNS was not smooth. For example, Sun Microsystems's Network Information System [CORP-SUN-NIS], at the time known as Yellow Pages, was an active competitor to the DNS, although it failed to provide a complete solution to the global naming problem.

Another example was NetBIOS Name Service, also known as WINS [RFC1002]. This protocol was used mostly by Microsoft Windows machines, but also by open source BSD and Linux operating systems to do name resolution using Microsoft's own name resolution protocol.

Most modern operating systems can still use the '/etc/hosts' file for name resolution. Many still have a name service switch that can be configured on the host to resolve some domains using NIS or WINS. Most have the capability to resolve names using mDNS by recognizing the special meaning of the '.local' SUTLDN.

The Sun Microsystems model of having private domains within a corporate site, while supporting the global domain name system for off-site persisted even after the NIS protocol fell into disuse. Microsoft used to recommend that site administrators allocate a "private" top-level domain for internal use, and this practice was very much a part of the zeitgeist at the time. This attitude is the root of the widespread practice of simply picking an unused top-level domain and using it for experimental purposes, which persists even at the time of writing of this memo.

This history is being presented because discussions about special-use names in the IETF often come down to the question of why users of new name resolution protocols choose to use Domain names, rather than using some other naming concept that doesn't overlap with the namespace that, in modern times is, by default, resolved using the DNS.

The answer is that as a consequence of this long history of resolving Domain Names, Domain Names appear in a large variety of contexts in user interfaces applications programming interfaces. Any name that appears in such a context is a Domain Name. What this means is that Domain Names have a highly privileged status in deployed software. Proposals to change that will be almost possible to get adopted in a useful or consistent way. And since in most operating systems, mechanisms already exist for implementing special handling for some Domain Names, from a practical perspective the only way to achieve

the goals of any new name resolution protocol is through the use of special-use Domain Names.

6. Contributors

This document came about as a result of conversations that occurred in the lobby, the weekend before IETF 95. Stuart Cheshire, Mark Andrews, David Conrad, Paul Ebersman and Aaron Falk all made helpful and insightful observations or patiently answered questions. This should not be taken as an indication that any of these folks actually agree with what the document says, but their generosity with time and thought are appreciated in any case.

Ralph started out as an innocent bystander, but discussion with him was the key motivating factor in the writing of this document, and he agreed to co-author it without too much arm-twisting. Warren spent a lot of time working with me on this document after it was first published, and joined as an author in order to make sure that the work got finished; without him the -01 and -02 versions might not have happened.

And this document owes a great deal to Ed Lewis' excellent work on what a "domain name" is [I-D.lewis-domain-names].

7. Informative References

[RFC0882] Mockapetris, P., "Domain names: Concepts and facilities", RFC 882, DOI 10.17487/RFC0882, November 1983, <http://www.rfc-editor.org/info/rfc882>.

[RFC0883] Mockapetris, P., "Domain names: Implementation specification", RFC 883, DOI 10.17487/RFC0883, November 1983, <http://www.rfc-editor.org/info/rfc883>.

[RFC0959] Postel, J. and J. Reynolds, "File Transfer Protocol", STD 9, RFC 959, DOI 10.17487/RFC0959, October 1985, <http://www.rfc-editor.org/info/rfc959>.

[RFC1002] NetBIOS Working Group in the Defense Advanced Research Projects Agency, Internet Activities Board, and End-to-End Services Task Force, "Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications", STD 19, RFC 1002, DOI 10.17487/RFC1002, March 1987, <http://www.rfc-editor.org/info/rfc1002>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <http://www.rfc-editor.org/info/rfc1035>.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[RFC2826] Internet Architecture Board, "IAB Technical Comment on the Unique DNS Root", RFC 2826, DOI 10.17487/RFC2826, May 2000, <http://www.rfc-editor.org/info/rfc2826>.

[RFC2860] Carpenter, B., Baker, F., and M. Roberts, "Memorandum of Understanding Concerning the Technical Work of the Internet Assigned Numbers Authority", RFC 2860, DOI 10.17487/RFC2860, June 2000, <http://www.rfc-editor.org/info/rfc2860>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC6303] Andrews, M., "Locally Served DNS Zones", BCP 163, RFC 6303, DOI 10.17487/RFC6303, July 2011, <http://www.rfc-editor.org/info/rfc6303>.

[RFC6598] Weil, J., Kuarsingh, V., Donley, C., Liljenstolpe, C., and M. Azinger, "IANA-Reserved IPv4 Prefix for Shared Address Space", BCP 153, RFC 6598, DOI 10.17487/RFC6598, April 2012, <http://www.rfc-editor.org/info/rfc6598>.

[RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol to Replace the AppleTalk Name Binding Protocol (NBP)", RFC 6760, DOI 10.17487/RFC6760, February 2013, <http://www.rfc-editor.org/info/rfc6760>.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, DOI 10.17487/RFC6761, February 2013, <http://www.rfc-editor.org/info/rfc6761>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, DOI 10.17487/RFC7050, November 2013, <http://www.rfc-editor.org/info/rfc7050>.

[RFC7686] Appelbaum, J. and A. Muffett, "The ".onion" Special-Use Domain Name", RFC 7686, DOI 10.17487/RFC7686, October 2015, <http://www.rfc-editor.org/info/rfc7686>.

[I-D.chapin-additional-reserved-tlds] Chapin, L. and M. McFadden, "Additional Reserved Top Level Domains", draft-chapin-additional-reserved-tlds-02 (work in progress), March 2015.

[I-D.lewis-domain-names] Lewis, E., "Domain Names", draft-lewis-domain-names-03 (work in progress), June 2016.

[SDO-CABF-INT] CA/Browser Forum, "Guidance on the Deprecation of Internal Server Names and Reserved IP Addresses", June 2012, <https://www.digicert.com/internal-names.htm>.

[SDO-ICANN-COLL] Interisle Consulting Group, LLC, "Name Collisions in the DNS", August 2013, <https://www.icann.org/en/system/files/files/name-collision-02aug13-en.pdf>.

[SDO-IANA-SUDR] Internet Assigned Numbers Authority, "Special-Use Domain Names registry", October 2015, <http://www.iana.org/assignments/special-use-domain-names/special-use-domain-names.xhtml>.

[CORP-SUN-NIS] Sun Microsystems, "Large System and Network Administration", March 1990.

[IETF-PRO-51] Internet Engineering Task Force, "Proceedings of the 51st IETF", August 2001, <http://www.ietf.org/proceedings/51/51-45.htm#TopOfPage>.

Appendix A. Change Log.

-03 to -04:

- Replaced 'Internet Names' with 'Domain Names' - suggestion by John Levine.

-02 to -03:

- Readability fixes, small grammar updates.

-01 to -02:

- Cleaned up the abstract.
- Fixed the case of Internet
- Reference to Ed Lewis' "domain names"
- Fleshed out the problems, primarily the coordination, collisions ones.

-00 to -01:

- Large refactoring, basically a rewrite. Incorporated comments, removed a bunch of unneeded text, etc.

Authors' Addresses

Ted Lemon
Nominum, Inc.
800 Bridge Parkway
Redwood City, California 94065
United States of America

Phone: +1 650 381 6000
Email: ted.lemon@nominum.com

Ralph Droms
Cisco, Inc.

Email: rdroms.ietf@gmail.com

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

dnsop
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

W. Kumari
Google
Z. Yan
CNNIC
W. Hardaker
USC/ISI
D. Lawrence
Akamai Technologies

July 3, 2017

Returning extra answers in DNS responses. draft-wkumari-dnsop-multiple-responses-05

Abstract

This document (re)introduces the ability to provide multiple answers in a DNS response. This is especially useful as, in many cases, the entity making the request has no a prori knowledge of what other questions it will need to ask.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Table of Contents

1. Introduction

In many cases a name being resolved in the DNS provides the reason behind why the name is being resolved. This may allow the authoritative nameserver to predict what other answers a recursive resolver will soon query for. By providing multiple answers in the response, the authoritative name server operator can assist a caching recursive resolver in pre-populating its cache before a stub resolver or other client asks for the subsequent queries. Apart from decreasing the latency for end users [RFC6555], this also decreases the total number of queries that the recursive resolver needs to send and the authoritative server needs to answer.

For example, the domain name administrator of Example Widgets, Inc (example.com) knows that the web page at www.example.com contains various other resources, including some images (served from images.example.com), some Cascading Style Sheets (served from css.example.com) and some JavaScript (served from data.example.com). An application attempting to resolve www.example.com is very likely to be a web browser rendering the page and will likely also need to resolve all of these additional names as well. Providing all of these answers in response to a query for www.example.com allows the recursive resolver to pre-populate its cache and have these answers

available immediately when a stub resolver or other DNS client asks for them. What is important to notice here is that the stub resolver does not know what other questions it will need to make until after it has already made the request for www.exmaple.com, received the reply, made the HTTP connection and parsed the HTML.

Other examples where this technique may be useful include SMTP (by including mail server addresses, SPF and DKIM records when serving the MX record), SRV (by providing the target information in addition to the SRV response) and TLSA (by providing any TLSA records associated with a name). This same technique can also be used to include both the IPv4 (A) and IPv6 (AAAA) addresses for any singular address query.

This technique, described in this document, is purely an optimization and enables a zone publisher to distribute other related answers that the client is likely to need along with an answer to the original request. Users get a better experience, recursive resolvers need to send less queries, authoritative servers have to answer fewer queries, etc.

1.1. Requirements notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Background

The existing DNS specifications [RFC1034] allow for supplemental information to be included in the "additional" section of the DNS response, but in order to defeat cache poisoning attacks most implementations either ignore or don’t trust additional records they didn’t ask for. For more background, see [Ref.Bellovin] and [RFC2181].

Not trusting the information in the additional section was necessary since there was no way to authenticate it. If a resolver queries for www.example.com and received answers for www.invalid.com as well, it is impossible for a non-validating resolver to tell if these were actually from invalid.com or if an attacker was trying to push bad information for invalid.com into the resolver’s cache. In a world of ubiquitous DNSSEC deployment [Ed note: By the time this document is published, there *will* be ubiquitous DNSSEC :-)], a validating resolver can validate, authenticate and trust the records in the additional information.

3. Terminology

Additional records Additional records are records that the authoritative nameserver has included in the Additional section.

EXTRA Resource Record The EXTRA resource record (defined below) carries a list of additional records to send.

Primary query A Primary query (or primary question) is a QNAME that the name server operator would like to return additional answers for.

Supporting DNSSEC information Supporting DNSSEC information is the DNSSEC RRSIGs that prove the authenticity of the Additional records.

Stub Resolver The term "Stub Resolver" is used in this document to refer to the most common instance of a DNS client sending DNS requests to a Recursive Resolver. However, other DNS clients are not excluded from these usages and where we write "Stub Resolver" you may read it as "Stub Resolver or other DNS client".

4. Returning multiple answers

The authoritative nameserver should include as many of the instructed additional records identified by the Extra Resource Record and Supporting DNSSEC information as will fit in the response packet. These additional records (and Supporting DNSSEC information) are appended to the additional section of the response.

In order to include additional records in a response, these conditions need to be met:

1. Additional records MUST only be included when the Name Server is authoritative for the zone, and the records to be returned are DNSSEC signed.

2. The supporting DNSSEC information necessary to perform validation on the records MUST be included.

3. The Authoritative Name Server SHOULD include as many of the additional records as will fit in the response. Additional records SHOULD be inserted in the order specified in the Additional records list.

4. Zone administrators SHOULD only include records identified in the EXTRA Resource Records that they expect a client to need.

5. The EXTRA Resource Record

To allow a zone content administrator to instruct the name server which additional records to serve when it receives a query to a label, we introduce the EXTRA Resource Record (RR). These additional records are appended to the additional section (note that the EXTRA RR itself is not appended). The EXTRA resource record MAY still be queried for directly (e.g for debugging), in which case the record itself is returned.

5.1. File Format

The format of the EXTRA RR is:

label EXTRA "label,type; label,type; label,type; ..."

For example, if the operator of example.com would like to also return A record answers for images.example.com, css.html.example.com and both an A and AAAA for data.example.com when queried for www.example.com, they would create the following record:

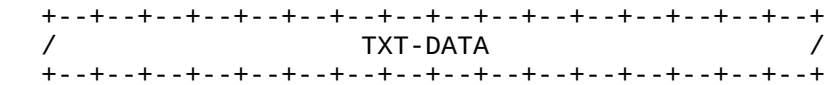
www.example.com. EXTRA "images,A; css,A; data,A; data,AAAA;"

The entries in the EXTRA list are ordered. An authoritative nameserver SHOULD insert the records in the order listed when filling the response packet. This is to allow the operator to express a preference in case all the records will not fit in the response. The TTL of the records added to the Additional section are MUST be the same as if queried directly.

In some cases a zone content administrator might not know what all additional records clients need. For example, the owner of www.example.com may have outsourced his DNS operations to a third party, and / or the DNS operator might not interact with the web development team. DNS server operators may use tools to mine their query logs for records to include. For example, if, in a large majority of cases, a recursive server asks for foo.example.com and then very soon after asks for bar.example.com, it may make sense to optimize this by opportunistically returning bar when queried for foo. This functionality could also be included in the authoritative name server software itself. The exact mechanisms and heuristics used for this are not discussed in this document.

5.2. Wire Format

The wire format of the EXTRA RR is the same as the wire format for a TXT RR:



Where TXT-DATA is one or more <character-string>s.

The EXTRA RR has RR type TBD [RFC Editor: insert the IANA assigned value and delete this note]

6. Signaling support

Recursive Resolvers (or other DNS clients) that support EXTRA records MAY signal this by setting the OPT record's EXTRA bit (bit NN [TBD: assigned by IANA] in the EDNS0 extension header to 1).

7. Stub-Resolver Considerations

No modifications need to be made to stub-resolvers to get the predominate benefit of this protocol, since the majority of the speed gain will take place between the validating recursive resolver and the authoritative name server. However, stub resolvers may choose to support this technique, and / or may query directly for the EXTRA RR if it wants to pre-query for data that will likely be needed in the process of supporting applications.

8. Use of Additional information

When deciding to use additional records in the additional section, a resolver must follow certain rules:

1. Additional records MUST be validated before being used.

2. Additional records SHOULD have lower priority in the cache than answers received because they were requested. This is to help evict Additional records from the cache first (to help prevent cache filling attacks).

3. Recursive resolvers MAY choose to ignore Additional records for any reason, including CPU or cache space concerns, phase of the moon, etc. It may choose to accept all, some or none of the Additional records.

9. IANA Considerations

This document contains the following IANA assignment requirements:

1. The EXTRA bit discussed in Section 6 needs to be allocated. [Ed: This section to be completed later]

10. Security Considerations

Additional records will make DNS responses even larger than they are currently, leading to larger records that can be used in DNS reflection attacks. One could mitigate this by only serving responses to EXTRA requests over TCP or when using Cookies [RFC5395], although there is no easy way to signal this to a client other than through the use of the truncate bit.

A malicious authoritative server could include a large number of extra records (and associated DNSSEC information) and attempt to DoS the recursive by making it do lots of DNSSEC validation. However, this is not considered a realistic threat; CPU for validation is cheap compared to bandwidth. This can be mitigated by allowing the recursive resolver to ignore Additional records whenever it considers itself under attack or its CPU resources are otherwise over-committed.

This specification requires that the all of the Additional records are signed, and all necessary DNSSEC information for validation be included to avoid cache poisoning attacks.

11. Acknowledgements

The authors to thank Mark Andrews, John Dickinson, Kazunori Fujiwara, Bob Harold, John Heidemann, and Tony Finch. The authors apologize in advance for others who contributed, but who we managed to forget.

12. Normative References

[Ref.Bellovin] Bellovin, S., "Using the Domain Name System for System Break-Ins", 1995, <https://www.cs.columbia.edu/~smb/papers/dnshack.pdf>.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <http://www.rfc-editor.org/info/rfc2181>.

[RFC5395] Eastlake 3rd, D., "Domain Name System (DNS) IANA Considerations", RFC 5395, DOI 10.17487/RFC5395, November 2008, <http://www.rfc-editor.org/info/rfc5395>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC7873] Eastlake 3rd, D. and M. Andrews, "Domain Name System (DNS) Cookies", RFC 7873, DOI 10.17487/RFC7873, May 2016, <http://www.rfc-editor.org/info/rfc7873>.

Appendix A. Changes / Author Notes.

[RFC Editor: Please remove this section before publication]

From -04 to -05:

- In the deadline rush, Warren forgot to add Tale. Fixed.
- Some more text fixups and clarifications.

From -03 to -04:

- Some additional text explaining how this differs from solutions which include multiple queries (you don’t know what to ask until you have received some answers).

From -02 to -03:

- Sat down and rewrote and cleaned up large sections of text.
- Changed name of RR from Additional to EXTRA (the term "Additional" is overloaded in general)
- Clarified that stub resolvers and other clients MAY use this specification.
- Attempted to clarify that the individual RRs are added to the response, not the EXTRA record itself. The EXTRA RR can be queried directly.

From -00 to -01:

- Nothing change in the template.

Authors' Addresses

Warren Kumari
Google
1600 Amphitheatre Parkway
Mountain View, CA 94043
US

Email: warren@kumari.net

Zhiwei Yan
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing 100190
P. R. China

Email: yanzhiwei@cnnic.cn

Wes Hardaker
USC/ISI
P.O. Box 382
Davis, CA 95617
US

Email: ietf@hardakers.net

David C Lawrence
Akamai Technologies
150 Broadway
Cambridge, MA 02142-1054
US

Email: tale@akamai.com

DNSOP Working Group
Internet-Draft
Updates: 2308, 4033, 4034, 4035 (if approved)
Intended status: Informational
Expires: January 22, 2020

J. Woodworth
D. Ballew
CenturyLink, Inc.
S. Bindinganaveli Raghavan
Hughes Network Systems
D. Lawrence
Oracle

July 21, 2019

BULK DNS Resource Records
draft-woodworth-bulk-rr-09

Abstract

The BULK DNS resource record type defines a method of pattern-based creation of DNS resource records based on numeric substrings of query names. The intent of BULK is to simplify generic assignments in a memory-efficient way that can be easily shared between the primary and secondary nameservers for a zone.

Ed note

Text inside square brackets ([]) is additional background information, answers to frequently asked questions, general musings, etc. They will be removed before publication. This document is being collaborated on in GitHub at <https://github.com/ioneyez/bulk-rr>. The most recent version of the document, open issues, etc should all be available here. The authors gratefully accept pull requests.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 22, 2020.

Table of Contents

1. Introduction

The BULK DNS resource record defines a pattern-based method for on-the-fly resource record generation. It is essentially an enhanced wildcard mechanism, constraining generated resource record owner names to those that match a pattern of variable numeric substrings. It is also akin to the $GENERATE master file directive [bind-arm] without being limited to numeric values and without creating all possible records in the zone data.

For example, consider the following record:

```
example.com. 86400 IN BULK A (
                      pool-A-[0-255]-[0-255].example.com.
                      10.55.${1}.${2}
                    )
```

It will answer requests for pool-A-0-0.example.com through pool-A-255-255.example.com with the IPv4 addresses 10.55.0.0 through 10.55.255.255.

Much larger record sets can be defined while minimizing the associated requirements for server memory and zone transfer network bandwidth.

This record addresses a number of real-world operational problems that authoritative DNS service providers experience. For example, operators who host many large reverse lookup zones, even for only IPv4 space in in-addr.arpa, would benefit from the disk space, memory size, and zone transfer efficiencies that are gained by encapsulating a simple record-generating algorithm versus enumerating all of the individual records to cover the same space.

Production zones of tens of thousands of pattern-generated records currently exist, that could be reduced to just one BULK RR. These zones can look deceptively small on the primary nameserver and balloon to 100MB or more when expanded,

BULK also allows administrators to more easily deal with singletons, records in the pattern space that are an exception to the normal data generation rules. Whereas a mechanism like $GENERATE may need to be adjusted to account for these individual records, the processing rules for BULK have explicit records more naturally override the dynamically generated ones. This collision problem is not just a

theoretical concern, but a real source of support calls for providers.

Pattern-generated records are also not only for the reverse DNS space. Forward zones also occasionally have entries that follow patterns that would be well-addressed by the BULK RR.

1.1. Background and Terminology

The reader is assumed to be familiar with the basic DNS and DNSSEC concepts described in [RFC1034], [RFC1035], [RFC4033], [RFC4034], and [RFC4035]; subsequent RFCs that update them in [RFC2181] and [RFC2308]; and DNS terms in [RFC7719].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when, and only when, they appear in all capitals, as shown here.

2. The BULK Resource Record

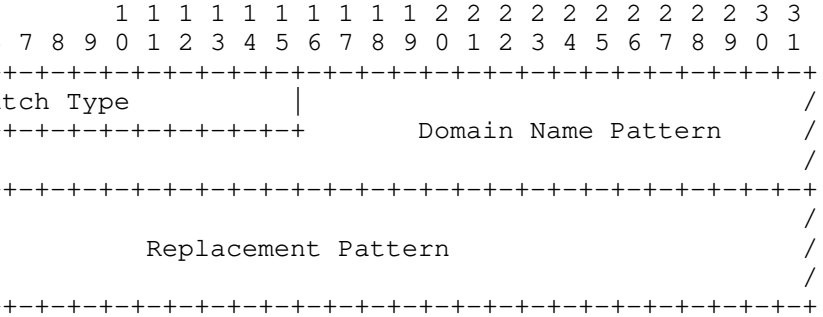
The BULK resource record enables an authoritative nameserver to generate RRs for other types based upon the query received.

The Type value for the BULK RR type is TBD.

The BULK RR is class-independent.

2.1. BULK RDATA Wire Format

The RDATA for a BULK RR is as follows:



Match Type identifies the type of the RRset to be generated by this BULK record. It is two octets corresponding to an RR TYPE code as specified in [RFC1035], Section 3.2.1.

Domain Name Pattern consists of a pattern encoded as a wire-format fully qualified domain name. The full name is used so that numeric substrings above the zone cut can be captured in addition to those in the zone. It needs no length indicator for the entire field because the root label marks its end.

Special characters are interpreted as per the following Augmented Backus-Naur Form (ABNF) notation from [RFC5234].

```
match         =  1*(range / string)

range         =  "[" [decnum "-" decnum] "]" /
                  "<" [hexnum "-" hexnum] ">"
                       ; create references for substitution
                       ; limit of 32 references
                       ; [] is syntactic sugar for 0-255
                       ; <> is syntactic sugar for 00-ff

string        =  1*(ctext / quoted-char)

decnum        =  1*decdigit
                       ; constrained to 65535 maximum.

hexnum        =  1*hexdigit
                       ; constrained to ffff maximum.

octet         =  %x00-FF

decdigit      =  %x30-39
                       ; 0-9
hexdigit      =  decdigit / 0x41-0x46 / 0x61-66
                       ; 0-9, A-F, a-f

ctext         =  <any octet excepting "\">

quoted-char   = "\" octet
                        ; to allow special characters as literals
```

Interpretation of the Domain Name Pattern is described in detail in the "BULK Replacement" section. Note that quoted-char must be stored in the wire format to preserve its semantics when the BULK RR is interpreted by nameservers.

The limit of 32 references is meant to simplify implementation details. It is largely but not entirely arbitrary, as it could capture every individual character of the text representation of a full IPv6 address.

Replacement Pattern describes how the answer RRset MUST be generated for the matching query. It needs no length indicator because its end can be derived from the RDATA length minus Match Type and Domain Name Pattern lengths. It uses the following additional ABNF elements:

```
replace       =   1*(reference / string)

reference     =   "$" "{" (positions / "*") [options] "}"

positions     =   (position / posrange) 0*("," (position / posrange))

posrange      =   position "-" position

position      =   1*decnum

options       =   delimiter [interval [padding]]

delimiter     =   "|" 0*(ctext | quoted-char)
                        ; "\|" to use "|" as delimiter
                        ; "\\" to use "\" as delimiter

interval      =   "|" *2decdigit

padding       =   "|" *2decdigit
```

[Is the formatting complexity beyond simple ${1}, ${2}, etc, really worth it? I definitely see how it could make for shorter replacement patterns, but does it enhance their clarity and usability, adding a feature someone really wants?]

The Replacement Pattern MUST end in the root label if it is intended to represent a fully qualified domain name.

2.2. The BULK RR Presentation Format

Match Type is represented as an RR type mnemonic or with [RFC3597]'s generic TYPE mechanism.

Domain Name Pattern is represented as a fully qualified domain name as per [RFC1035] Section 5.1 rules for encoding whitespace and other special characters.

Replacement Pattern is represented by the standard <character-string> text rules for master files as per [RFC1035] section 5.1.

It is suggested that lines longer than 80 characters be wrapped with parenthetical line continuation, per [RFC1035] Section 5.1, starting after Match Type and ending after Replacement Pattern.

3. BULK Replacement

When a BULK-aware authoritative nameserver receives a query for which it does not have a matching name or a covering wildcard, it MUST then look for BULK RRs at the zone apex, selecting all BULK RRs with a Match Type that matches the query type and a Domain Name Pattern that matches the query name. Note that query type ANY will select all Match Types, and all query types match a CNAME or DNAME Match Type. One or more answer RRs will be generated per the replacement rules below. Examples are provided in an appendix.

By only triggering the BULK algorithm when the query name does not exist, administrators are given the flexibility to explicitly override the behaviour of specific names that would otherwise match the BULK record's Domain Name Pattern. This is unlike BIND's $GENERATE directive, which adds the generated RRs to any existing names.

3.1. Matching the Domain Name Pattern

A query name matches the Domain Name Pattern if the characters that appear outside the numeric ranges match exactly and those within numeric ranges have values that fall within the range. Numeric matches MUST be of the appropriate decimal or hexadecimal type as specified by the delimiters in the pattern. For example, if a range is given as [0-255], then FF does not match even though its value as a hexadecimal number is within the range. Leading zeros in the numeric part(s) of the qname MUST be ignored; for example, 001.example.com, 01.example.com and 1.example.com would all match [].example.com.

When a query name matches a Domain Name Pattern, the value in each numeric range is stored for use by the Replacement Pattern, with reference numbers starting at 1 and counting from the left. For example, matching the query name host-24-156 against host-[0-255]-[0-255] assigns 24 to ${1} and 156 to ${2}.

3.2. Record Generation using Replacement Pattern

The Replacement Pattern generates the record data by replacing the ${...} references with data captured from the query name, and copying all other characters literally.

The simplest form of reference uses only the reference number between the braces, "{" and "}". The value of the reference is simply copied directly from the matching position of the query name.

The next form of reference notation uses the asterisk, "*". With ${*}, all captured values in order of ascending position, delimited by its default delimiter (described below), are placed in the answer. The commercial-at, "@" symbol captures in the same way only in order of descending position.

Numeric range references, such as ${1-4}, replaces all values captured by those references, in order, delimited by the default delimiter described below. To reverse the order in which they are copied, reverse the upper and lower values, such as ${4-1}. This is useful for generating PTR records from query names in which the address is encoded in network order.

Similar to range references, separating positions by commas creates sets for replacement. For example, ${1,4} would be replaced by the first and fourth captured values, delimited its default delimiter. This notation may be combined with the numeric range form, such as ${3,2,1,8-4}.

3.2.1. Delimiters

A reference can specify a delimiter to use by following a vertical bar, "|", with zero or more characters. Zero characters, such as in ${1-3|}, means no delimiter is used, while other characters up to an unescaped vertical bar or closing brace are copied between position values in the replacement. The default delimiter is the hyphen, "-".

3.2.2. Delimiter intervals

A second vertical bar in the reference options introduces a delimiter interval. The default behavior of a multi-position reference is to combine each captured value specified with a delimiter between each. With a delimiter interval the delimiters are only added between every Nth value. For example, ${*|-|4} adds a hyphen between every group of four captured positions. This can be a handy feature in the IPv6 reverse namespace where every nibble is captured as a separate value and generated hostnames include sets of 4 nibbles. An empty or 0 value for the delimiter interval MUST be interpreted as the default value of 1.

3.2.3. Padding length

The fourth and final reference option determines the field width of the copied value. Shorter values MUST be padded with leading zeroes ("0") and longer values MUST be truncated to the width.

The default behavior, and that of an explicit empty padding length, is that the captured query name substring is copied exactly. A width of zero "0" is a signal to "unpad", and any leading zeros MUST be removed. [Unnecessary complexity?]

If a delimiter interval greater than 1 is used, captured values between the intervals will be concatenated and the padding or unpadding applied as a unit and not individually. An example of this would be ${*||4|4} which would combine each range of 4 captured values and pad or truncate them to a width of 4 characters.

[If this is kept, the element/feature should probably be renamed from "padding" since it is just as likely to truncate.]

3.2.4. Final processing

The string that results from all replacements is converted to the appropriate RDATA format for the record type. If the conversion fails, the SERVFAIL rcode MUST be set on the response, representing a misconfiguration that the server was unable to perform. [The EDNS extended-error code would be useful here.]

The TTL of each RR generated by a BULK RR is the TTL of the corresponding BULK record itself. [BULK should probably have its own TTL field because using that of the record itself feels like bad design. On the other hand, if BULK is never meant to be queried for directly and only appears in authoritative data, its own TTL is pretty useless normally.]

The class for the RRSet is the class of the BULK RR.

If the generated record type is one that uses domain names in its resource record data, such as CNAME, a relative domain names MUST be fully qualified with the origin domain of the BULK RR.

4. Known Limitations

This section defines known limitations of the BULK resource type.

4.1. Unsupported Nameservers

Authoritative nameservers that do not understand the semantics of the new record type will not be able to deliver the intended answers even when the type appears in their zone data This significantly affects the interoperability of primary versus secondary authorities that are not all running the same software. Adding new RRs which affect handling by authoritative servers, or being unable to add them, is an issue that needs to be explored more thoroughly within dnsop.

5. Security Considerations

Two known security considerations exist for the BULK resource record, DNSSEC and DDOS attack vectors.

5.1. DNSSEC Signature Strategies

DNSSEC was designed to provide validation for DNS resource records, requiring each tuple of owner, class, and type to have its own signature. This essentially defeats the purpose of providing large generated blocks of RRs in a single RR as each generated RR would require its own legitimate RRSIG record.

In the following sections several options are discussed to address this issue. Of the options, on-the-fly provides the most secure solution and NPN [npn-draft] provides the most flexible.

5.1.1. On-the-fly Signatures

A significant design goal of DNSSEC was to be able to do offline cryptographic signing of zone contents, keeping the key material more secure.

On-the-fly processing requires authoritative nameservers to sign generated records as they are created. Not all authoritative nameserver implementations offer on-the-fly signatures, and even with those that do not all operators will want to keep signing keys online. This solution would either require all implementations to support on-the-fly signing or be ignored by implementations which can not or will not comply.

One possibly mitigation for addressing the risk of keeping the zone signing key online would be to continue to keep the key for signing positive answers offline and introduce a second key for online signing of negative answers.

No changes to validating resolvers is required to support this solution.

5.1.2. Alternative Signature Scheme

Previous versions of this draft proposed a new signature scheme using a Numeric Pattern Normalization (NPN) RR. It was a method to support offline signatures for BULK records, with the drawback that is required updates to DNSSEC-aware resolvers.

That mechanism is not specific to BULK and has been removed from the current draft. If there is further interest in pursuing it, it can be reopened as a separate draft.

5.1.3. Non-DNSSEC Zone Support Only

As a final option zones which wish to remain entirely without DNSSEC support may serve such zones without either of the above solutions and records generated based on BULK RRs will require zero support from recursive resolvers.

5.2. DDOS Attack Vectors and Mitigation

As an additional defense against Distributed Denial Of Service (DDOS) attacks against recursive (resolving) nameservers it is highly recommended shorter TTLs be used for BULK RRs than others. While disabling caching with a zero TTL is not recommended, as this would only result in a shift of the attack target, a balance will need to be found. While this document uses 24 hours (86400 seconds) in its examples, values between 300 to 900 seconds are likely more appropriate and is RECOMMENDED. What is ultimately deemed appropriate may differ from zone to zone and administrator to administrator.

[I am unclear how this helps DDOS mitigation against anyone at all, and suspect this section should be removed..]

5.3. Implications of Large-Scale DNS Records

The production of such large-scale records in the wild may have some unintended side-effects. These side-effects could be of concern or add unexpected complications to DNS based security offerings or forensic and anti-spam measures. While outside the scope of this document, implementers of technology relying on DNS resource records for critical decision making must take into consideration how the existence of such a volume of records might impact their technology.

Solutions to the magnitude problem for BULK generated RRs are expected be similar if not identical to that of existing wildcard records, the core difference being the resultant RDATA will be unique for each requested Domain Name within its scope.

The authors of this document are confident that by careful consideration, negative_side-effects produced by implementing the features described in this document can be eliminated from any such service or product.

6. Privacy Considerations

The BULK record does not introduce any new privacy concerns to DNS data.

7. IANA Considerations

IANA is requested to assign numbers for the BULK RR.

8. Acknowledgments

This document was created as an extension to the DNS infrastructure. As such, many people over the years have contributed to its creation and the authors are appreciative to each of them even if not thanked or identified individually.

A special thanks is extended for the kindness, wisdom and technical advice of Robert Whelton (CenturyLink, Inc.) and Gary O'Brien (Secure64 Software Corp).

9. References

9.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <https://www.rfc-editor.org/info/rfc2181>.

[RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <https://www.rfc-editor.org/info/rfc2308>.

[RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <https://www.rfc-editor.org/info/rfc2317>.

[RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <https://www.rfc-editor.org/info/rfc3597>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

9.2. Informative References

[bind-arm] Internet Systems Consortium, "BIND 9 Configuration Reference", 2016, <https://ftp.isc.org/isc/bind9/cur/9.9/doc/arm/Bv9ARM.html>.

[npn-draft] Internet Systems Consortium, "Numeric Pattern Normalization (NPN)", 2019, <https://github.com/ioneyez/npn>.

[RFC7719] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", RFC 7719, DOI 10.17487/RFC7719, December 2015, <https://www.rfc-editor.org/info/rfc7719>.

Appendix A. BULK Examples

A.1. Example 1

```
$ORIGIN 2.10.in-addr.arpa.
@ 86400 IN BULK PTR (
          [0-255].[0-255].[0-255].[0-255].in-addr.arpa.
          pool-${4-1}.example.com.
        )
```

A query received for the PTR of 4.3.2.10.in-addr.arpa will create the references ${1} through ${4} with the first four labels of the query name. The ${4-1} reference in the replacement pattern will then substitute them in reverse with the default delimiter of hyphen between every character and no special field width modifications. The TTL of the BULK RR is used for the generated record, making the response:

4.3.2.10.in-addr.arpa 86400 IN PTR pool-10-2-3-4.example.com.

A.2. Example 2

```
$ORIGIN 2.10.in-addr.arpa.
@ 86400 IN BULK PTR (
          [0-255].[0-255].[0-255].[0-255].in-addr.arpa.
          pool-${2,1|||3}.example.com.
        )
```

Example 2 is similar to Example 1, except that it modifies the replacement pattern. The empty option after the first vertical bar causes no delimiters to be inserted, while the second empty option that would keep the delimiter interval as 1. The latter is relevant because the final value, padding of 3, is applied over each delimiter interval even when no delimiter is used. Not all captures from the substring are required to be used in the response.

The result is that a query for the PTR of 4.3.2.10.in-addr.arpa generates this response:

4.3.2.10.in-addr.arpa 86400 IN PTR pool-003004.example.com.

[Admittedly you can’t do this very effectively without the field width complexity. Is this sort of name common? Does it need

support? Admittedly $GENERATE had the feature, but is that reason enough?]

[Change this to a hex matching example?]

A.3. Example 3

```
$ORIGIN 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
@ 86400 IN BULK PTR (
          <>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>.<>
          poolAA-${16-8|-|4}.example.com.
        )
```

This example introduces IPv6 where 16 individual nibbles are captured and the last 8 are combined into 2 blocks of 4, separated by a hyphen.

A query for the IP of 2001:db8::dead:beef results in a PTR RR with the value of poolAA-dead-beef.example.com.

A.4. Example 4

```
$ORIGIN example.com.
@ 86400 IN BULK AAAA (
          poolAA-<0-ffff>-<0-ffff>.example.com.
          ${@|.|1}.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.
        )
```

This example performs the reverse of example 3, where a query of poolAA-dead-beef.example.com captures "dead" and "beef", reversing the nibbles and using a dot (.) as the delimiter to form a valid AAAA record.

A.5. Example 5

This example contains a classless IPv4 delegation on the /22 CIDR boundary as defined by [RFC2317]. The network for this example is "10.2.0/22" delegated to a nameserver "ns1.sub.example.com.". RRs for this example are defined as:

```
$ORIGIN 2.10.in-addr.arpa.
@    7200 IN BULK CNAME [0-255].[0-3] ${*|.}.0-3
0-3 86400 IN NS ns1.sub.example.com.
```

A query for the PTR of 25.2.2.10.in-addr.arpa is received and the BULK record with the CNAME Match Type matches all query types. 25 and 2 are captured as references, and joined in the answer by the period (".") character as a delimiter, with ".0-3" then appended

literally and fully qualified by the origin domain. The final synthesized record is:

25.2.2.10.in-addr.arpa 7200 IN CNAME 25.2.0-3.2.10.in-addr.arpa.

[Without $* and options complexity, the pattern to get the same result is just ${1}.{$2}.0-3 which is not really significantly onerous to enter, and slightly less arcane looking to comprehend.]

Authors' Addresses

John Woodworth
CenturyLink, Inc.
4250 N Fairfax Dr
Arlington VA 22203
USA

Email: John.Woodworth@CenturyLink.com

Dean Ballew
CenturyLink, Inc.
2355 Dulles Corner Blvd, Ste 200 300
Herndon VA 20171
USA

Email: Dean.Ballew@CenturyLink.com

Shashwath Bindinganaveli Raghavan
Hughes Network Systems
11717 Exploration Lane
Germantown MD 20876
USA

Email: shashwath.bindinganaveliraghavan@hughes.com

David C Lawrence
Oracle

Email: tale@dd.org

dnsop
Internet-Draft
Intended status: Standards Track
Expires: March 21, 2018

J. Yao
P. Vixie
CNNIC-Farsight Joint Laboratory
N. Kong
X. Li
CNNIC

September 17, 2017

A DNS Query including A Main Question with Accompanying Questions
draft-yao-dnsop-accompanying-questions-04

Abstract

This document enables DNS initiators to send a main question accompanying with several related questions in a single DNS query, and enables DNS responders to put the answers into a single DNS response. This extension enables a range of initiators to look up "X, or failing that, Y" in a better way than both current alternatives. This mechanism can reduce the number of DNS round-trips per application work-unit.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 21, 2018.

Table of Contents

1. Introduction

Sometimes, when DNS lookup of X, an application will lookup Y if X fails. For examples, the initiator may fall back to A record if the lookup of MX record fails.

Some initiators do it in sequence, X and after a few seconds, then Y. Although it is simple, this leads to unpleasant waiting whenever X times out or answers negatively.

Some initiators use concurrent X/Y lookups and a state machine to decide whether to use X or Y. If an answer to Y arrives but none to X, the initiator needs to wait a little or else fall back to Y inappropriately. Concurrent lookup is faster if the X lookup takes time and falling back to Y is appropriate, but rather complex, with four states to test, and the initiator needs to wait for an answer to X or a timeout before it can use Y.

This document enables a quicker, more easily tested failover. There is no need to test different answer sequences, there's no need for a state machine, there's no need for timeouts beyond receiving the reply. This document describes a method by which DNS initiators can send a main question accompanying with several related questions in a single DNS query, and enables DNS responders place all related answers into a single DNS response. This mechanism can reduce the number of DNS round-trips per application work-unit, by carrying several related queries in a single query transaction. It has the following advantages compared to other solutions.

- Compared to sequential lookups: It's roughly as simple, but much faster in case a fallback to Y is necessary.

- Compared to the concurrent mechanism: It is slightly faster (if the initiator needs to wait for an X timeout) and/or prevents inappropriate fallback (if the answer to X arrives too late), and it has a simpler state machine.

This mechanism can also be used in the scenarios when the application needs more records of the same domain name or its sub-domain name. For examples, when asking about a QTYPE=A RRset, a QTYPE=AAAA RRset may also be of use [RFC 5321]; When asking for some RRset of www.example.com about A and AAAA, records of a sub-domain name such as _443._tcp.www.example.com for TLSA may be of interest[RFC 6698].

2. Terminology

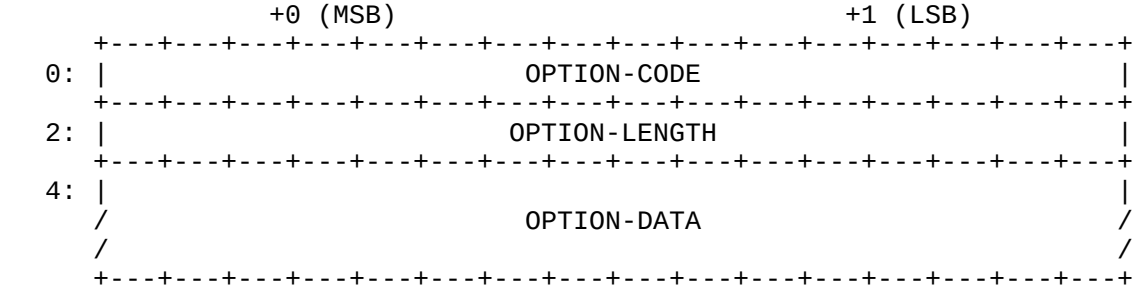
The basic key words such as "MUST", "MUST NOT", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "MAY", and "MAYNOT" are to be interpreted as described in [RFC2119].

The basic DNS terms used in this specification are defined in the documents [RFC1034] and [RFC1035].

3. Mechanism for a main question with accompanying questions

The initiator still puts a main question into the question section of the DNS query packet, as described in [RFC1035]. Accompanying questions will be put into the variable part of an OPT RR [RFC6891].

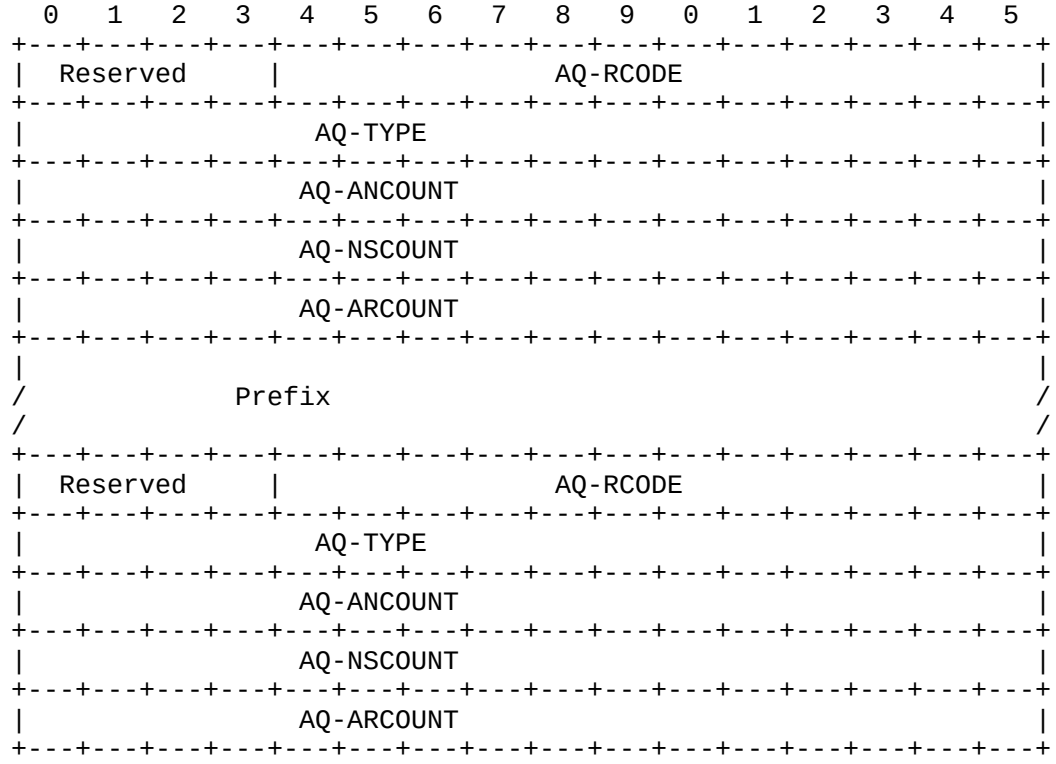
The variable part of an OPT RR is encoded in its RDATA and is structured as the following:

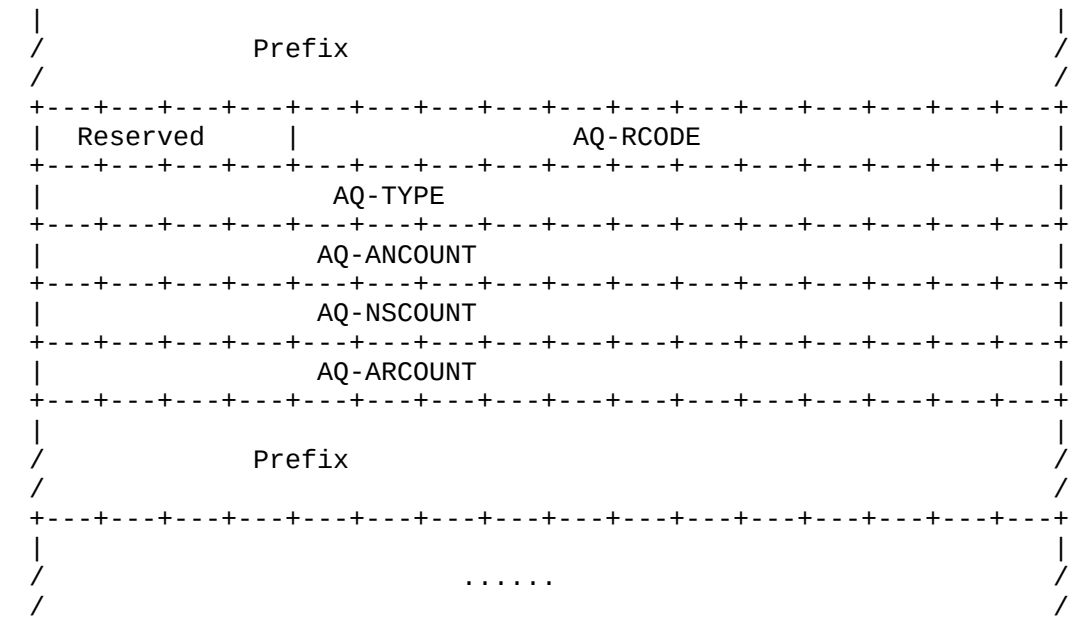


OPTION-CODE (Assigned by IANA.)

OPTION-LENGTH Size (in octets) of OPTION-DATA.

OPTION-DATA including at most 6 accompanying questions with AQ-RCODE.





- Reserved field is kept for the future use.

- AQ-RCODE field will be set to 111111110100 bits when being initialized. The AQ-RCODE with the value of 111111110100 bits means that the mechanism for accompanying has not been implemented, where "0100" in the RCODE value means "not been implemented". The AQ aware responders will put the RCODE value for the query of this question into AQ-RCODE fields.

- AQ-ANCOUNT field will indicate the number of resource records in the answer section for this accompanying question. The AQ aware responders will put the ANCOUNT value for the query of this question into AQ-ANCOUNT field.

- AQ-NSCOUNT field will indicate the number of name server resource records in the authority records section for this accompanying question. The AQ aware responders will put the NSCOUNT value for the query of this question into AQ-NSCOUNT field.

- AQ-ARCOUNT field will indicate the number of resource records in the additional records section for this accompanying question. The AQ aware responders will put the ARCOUNT value for the query of this question into AQ-ARCOUNT field.

- Prefix field indicates a domain name with the form of a dot or a sequence of labels ending with a pointer using the message compression defined in section 4.1.4. of RFC 1035. The domain name for accompanying questions MUST be same with the domain name for a main question or be children name of it. For an example, if the main domain name is example.com and the accompanying domain name is mail.example.com., the prefix is "mail." ending with a pointer pointing to "example.com.".

4. Responder Processing

The AQ aware responder will check the main question first, and put the results into the DNS response packet following RFC 1034. If the AQ OPT is present, the responder assembles the prefix with the main domain name and makes it to be an accompanying question, checks the accompanying questions in order, and put the results into the DNS answer section, authority section or additional records section of the response following RFC 1034; but the response code is placed in the respective AQ-RCODE field in AQ OPT of the response. The RCODE field in the DNS response header refers to the main question only. The AQ aware responders will put the ANCOUNT, NSCOUNT and ARCOUNT value for the query of this accompanying question into the respective AQ-ANCOUNT, AQ-NSCOUNT and AQ-ARCOUNT fields. The ANCOUNT, NSCOUNT and ARCOUNT fields in the DNS response header refer to the main question and its accompanying questions. Since the value for the accompanying questions' ANCOUNT, NSCOUNT and ARCOUNT can be known from the respective value of AQ-ANCOUNT, AQ-NSCOUNT and AQ-ARCOUNT, the actual value of the main question's ANCOUNT, NSCOUNT and ARCOUNT can be calculated from the ANCOUNT, NSCOUNT and ARCOUNT in the DNS response header. When the answer is negative for the accompanying question, the SOA resource record will be put in the authority section.

The mechanism proposed in this document is intended for both between stub resolvers and recursive resolvers, and between recursive resolvers and authoritative servers. If some DNS resource records are needed to be processed at the same time, the DNS administrator may configure it together. In case of that some children domain names are delegated and not in the main domain name's zone, the delegation information will be returned to the recursive resolvers. The recursive resolvers then check the children domain based on the delegation information, and get the answer for the respective children domain names.

When a stub resolver sends an AQ query to the recursive resolver, the recursive resolver may have some answers for one or more questions in the cache, but not for all questions. Under that case, the recursive resolver SHOULD forward this AQ query to some relative authoritative

servers for full answers instead of using the existing insufficient cache information.

An AQ unaware responder is expected to ignore the AQ OPT of the query, and may echo the received OPT back into additional section of the response message.

5. Initiator Processing

An AQ aware initiator will put the main question into the question section of the DNS query packet, and put each accompanying question into the related accompanying question fields of OPTION-DATA of OPT RR. AQ-RCODE value will be sent as 111111110100 bits. The AQ-TYPE value should be set as the query type related to accompanying questions. The Prefix value should be set as a dot or a sequence of labels ending with a pointer pointing to the the main domain name of the main question for the respective accompanying domain name of the accompanying question.

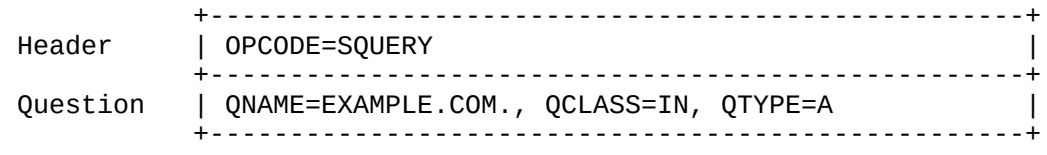
An AQ aware initiator SHOULD set the limitation of what is the maximum number of accompanying questions a AQ query can bring. This document suggests that the maximum number is six since most DNS resource records which need parallel query will not larger than six. The implementers may set six as the defaul value in the implementation. The responder can refuse to answer the AQ query if the maximum number of the accompanying questions is larger than the default maximum value, and return "not been implemented, too many accompanying-questions." information to the initiator.

If the initial value of the AQ-RCODE is unchanged in the response or the AQ OPT is not echo back, it indicates that the responder is AQ unaware. In that case, the responder will deal with the main question only. The initiator should sent the accompanying questions one by one via the normal DNS query. In such followup related queries, AQ processing should probably not be attempted, to reduce waste of network resources.

6. Query and Response Example

Example: one main question with 2 accompanying questions

The query would look like:



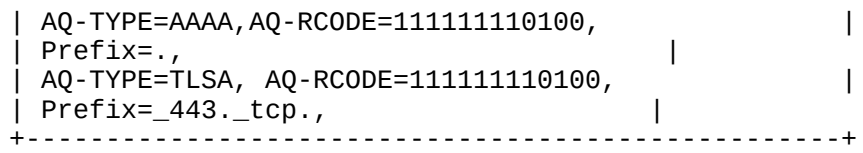
```
Answer     |                                                    |
           +----------------------------------------------------+
Authority  | <empty>                                            |
           +----------------------------------------------------+
Additional |                                                    |
           | AQ-TYPE=AAAA,AQ-RCODE=111111110100,                |
           | Prefix=.,                                  |
           | AQ-TYPE=TLSA,,AQ-RCODE=111111110100,               |
           | Prefix=_443._tcp.,                         |
           +----------------------------------------------------+
```

The response from AQ aware responders would be:

```
           +----------------------------------------------------+
Header     | OPCODE=SQUERY,  RESPONSE, AA, RCODE=NOERROR        |
           |      ANCOUNT=3, ARCOUNT=1, NSCOUNT=0               |
           +----------------------------------------------------+
Question   | QNAME=EXAMPLE.COM., QCLASS=IN, QTYPE=A             |
           +----------------------------------------------------+
Answer     |        example.com  IN A 192.168.0.1               |
           |        example.com. IN AAAA 2001:cc8::1            |
           |   _443._tcp.example.com. IN TLSA                   |
           |        ( 3 0 0 30820307308201efa003020102020... )  |
           +----------------------------------------------------+
Authority  | <empty>                                            |
           +----------------------------------------------------+
Additional |                                                    |
           | AQ-TYPE=AAAA, AQ-RCODE=NOERROR, AQ-ANCOUNT=1,      |
           |                    AQ-ARCOUNT=0, AQ-NSCOUNT=0,     |
           | Prefix=.,                                  |
           | AQ-TYPE=TLSA, AQ-RCODE=NOERROR, AQ-ANCOUNT=1,      |
           |                    AQ-ARCOUNT=0, AQ-NSCOUNT=0,     |
           | Prefix=_443._tcp.,                         |
           +----------------------------------------------------+
```

The response from AQ unaware responders would be:

```
           +----------------------------------------------------+
Header     | OPCODE=SQUERY,  RESPONSE, AA, RCODE=NOERROR        |
           +----------------------------------------------------+
Question   | QNAME=EXAMPLE.COM., QCLASS=IN, QTYPE=A             |
           +----------------------------------------------------+
Answer     |        example.com.  IN A 192.168.0.1              |
           +----------------------------------------------------+
Authority  | <empty>                                            |
           +----------------------------------------------------+
Additional |                                                    |
```



7. IANA Considerations

IANA should allocate DNS EDNS0 Option Codes (OPT) following this document. IANA should reserve RCODE with the value of 111111110100 bits for this document.

8. Security Considerations

TBD

9. Acknowledgements

The authors thank the members in DNSOP mailing list for helpful discussions, and especially thank Kazunori Fujiwara, JINMEI Tatuya, Bob Harold, Arnt Gulbrandsen, Olafur Gudmundsson and Stephane Bortzmeyer for kind comments, suggestions and improvements for the document. The authors also thanks Likun Zhang for helpful discussion about some topics related to implementation.

10. Change History

RFC Editor: Please remove this section.

10.1. draft-yao-dnsop-accompanying-questions: Version 00

- A Mechanism for DNS query including one main question with several accompanying questions

10.2. draft-yao-dnsop-accompanying-questions: Version 01

- Simpilfy the mechanism.

10.3. draft-yao-dnsop-accompanying-questions: Version 02

- Remove the AQ and Count bits, and add AQ-ANCOUNT AQ-ARCOUNT AQ-NSCOUNT

10.4. draft-yao-dnsop-accompanying-questions: Version 03

- Improve the introduction and explains the motivation of this draft

10.5. draft-yao-dnsop-accompanying-questions: Version 04

- Improve the document

11. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <https://www.rfc-editor.org/info/rfc1034>.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <https://www.rfc-editor.org/info/rfc5321>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <https://www.rfc-editor.org/info/rfc6698>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <https://www.rfc-editor.org/info/rfc6891>.

Authors' Addresses

Jiankang Yao
CNNIC-Farsight Joint Laboratory
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3007
Email: yaojk@cnnic.cn

Paul Vixie
CNNIC-Farsight Joint Laboratory
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +1 650 489 7919
Email: vixie@fsi.io

Ning Kong
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3147
Email: nkong@cnnic.cn

Xiaodong Li
CNNIC
4 South 4th Street,Zhongguancun,Haidian District
Beijing, Beijing 100190
China

Phone: +86 10 5881 3020
Email: xl@cnnic.cn

DNSOP
Internet-Draft
Intended status: Informational
Expires: April 24, 2019

D. York
Internet Society
O. Sury
ISC
P. Wouters
Red Hat
O. Gudmundsson
CloudFlare

October 21, 2018

Observations on Deploying New DNSSEC Cryptographic Algorithms
draft-york-dnsop-deploying-dnssec-crypto-algs-06

Abstract

As new cryptographic algorithms are developed for use in DNSSEC signing and validation, this document captures the steps needed for new algorithms to be deployed and enter general usage. The intent is to ensure a common understanding of the typical deployment process and potentially identify opportunities for improvement of operations.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 24, 2019.

Copyright Notice

Table of Contents

1. Introduction

The DNS Security Extensions (DNSSEC), broadly defined in [RFC4033], [RFC4034] and [RFC4035], make use of cryptographic algorithms in both the signing of DNS records and the validation of DNSSEC signatures by recursive resolvers.

The current list of cryptographic algorithms can be found in the IANA "Domain Name System Security (DNSSEC) Algorithm Numbers" registry located at <http://www.iana.org/assignments/dns-sec-alg-numbers/> Algorithms are added to this IANA registry through a process defined in [RFC6014]. Note that [RFC6944] provides some guidance as to which of these algorithms should be implemented and supported.

Historically DNSSEC signatures have primarily used cryptographic algorithms based on RSA keys. As deployment of DNSSEC has increased there has been interest in using newer and more secure algorithms, particularly those using elliptic curve cryptography.

The ECDSA algorithm [RFC6605] has seen some adoption and a new signing algorithm is now available: Edwards-curve Digital Signature Algorithm (EdDSA) using a choice of two curves, Ed25519 and Ed448, [RFC8080].

The challenge is that the deployment of a new cryptographic algorithm for DNSSEC is not a simple process. DNSSEC algorithms are used throughout the DNS infrastructure for tasks such as:

- Generation of keys ("DNSKEY" record) for signing
- Creation of DNSSEC signatures in zone files ("RRSIG")
- Usage in a Delegation Signer ("DS") record [RFC3658] for the "chain of trust" connecting back to the root of DNS
- Generation of NSEC/NSEC3 responses by authoritative DNS servers
- Validation of DNSSEC signatures by DNS resolvers

In order for a new cryptographic algorithm to be fully deployed, all aspects of the DNS infrastructure that interact with DNSSEC must be updated to use the new algorithm.

This document outlines the current understanding of the components of the DNS infrastructure that need to be updated to deploy a new cryptographic algorithm.

It should be noted that DNSSEC is not alone in complexity of deployment. The IAB documented "Guidelines for Cryptographic Algorithm Agility" in [RFC7696] to highlight the importance of this issue.

1.1. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Aspects of Deploying New Algorithms

For a new cryptographic algorithm to be deployed in DNSSEC, the following aspects of the DNS infrastructure must be updated:

- DNS resolvers performing validation
- Authoritative DNS servers

- Signing software
- Registries
- Registrars
- DNS Hosting Operators
- Applications

Each of these aspects is discussed in more detail below.

2.1. DNS Resolvers Performing Validation

DNS recursive resolvers perform "validation" to check the DNSSEC signatures of records received in a DNS query. To validate the signatures, the resolvers need to be able to understand the algorithm used to create the signatures.

In the case of a new algorithm, the resolver software needs to be updated. In some cases this could require waiting until an underlying library is updated to support the new algorithm.

Once the software is updated, the updates need to be deployed to all resolvers using that software. This can be challenging in cases of customer-premises equipment (CPE) that does not have any mechanism for automatic updating.

2.1.1. Resolvers and Unknown Algorithms

It should be noted that section 5.2 of [RFC4035] states:

"If the resolver does not support any of the algorithms listed in an authenticated DS RRset, then the resolver will not be able to verify the authentication path to the child zone. In this case, the resolver SHOULD treat the child zone as if it were unsigned."

This means that signing a zone with a new algorithm that is not widely supported by DNS resolvers would result in the signatures being ignored and the zone treated as unsigned until resolvers were updated to recognize the new algorithm.

Note that in at least one 2016 case the resolver software deployed on customer premises by an Internet service provider (ISP) turned out not to be compliant with RFC 4035. Instead of ignoring the signatures using unknown algorithms and treating the zones as unsigned, the validating resolver rejected the signatures and returned a SERVFAIL to the DNS query. This resulted in the ISP

turning off DNSSEC validation on the equipment. Further investigation showed that a newer version of the resolver software did correctly support ECDSA, but now all customer premises equipment must be updated to this new version.

The point is that it is not safe to assume all resolver software will correctly implement this part of RFC 4035.

2.2. Authoritative DNS Servers

Authoritative DNS servers serve out signed DNS records. Serving new DNSSEC signing algorithms should not be a problem as a well-written authoritative DNS server implementation should be agnostic to the RR DATA they serve.

The one exception is if the new cryptographic algorithms are used in the creation of NSEC/NSEC3 responses. In the case of new NSEC/NSEC3 algorithms, the authoritative DNS server software would need to be updated to be able to use the new algorithms.

Note that some authoritative server implementations could include DNSSEC signing as part of the server and thus also fall into the "Signing Software" category below.

2.3. Signing Software

The software performing the signing of the records needs to be updated with the new cryptographic algorithm.

User interfaces that allow users to interact with the DNSSEC signing software may also need to be updated to reflect the existence of the new algorithm.

Note that the key and signatures with the new algorithm will need to co-exist with the existing key and signatures for some period of time. This will have an impact on the size of the DNS records.

One issue that has been identified is that not all commonly-used signing software releases include support for an algorithm rollover. This software would need to be updated to support rolling an algorithm before any new algorithms could be deployed.

2.3.1. NSEC3 Iterations

[additional text needed]

2.4. Registries

The registry for a top-level domain (TLD) needs to accept DS records using the new cryptographic algorithm.

Observations to date have shown that some registries only accept DS records with certain algorithms. Registry representatives have indicated that they verify the accuracy of DS records to reduce technical support incidents and ensure customers do not mistakenly create any outages.

However, this means that registries who perform this level of checking must be able to understand new algorithms in order to successfully verify the DS records.

Separately, feedback from registrars has indicated that they do not currently have any mechanism to understand what DNSSEC algorithms a registry can accept.

2.5. Registrars

Registrars perform a critical role in the DNSSEC "chain of trust" of passing the DS record up to the Registry to ensure that the signed zone can be authenticated from the root of DNS all the way to the zone.

If the registrar is also providing the DNS hosting services for a domain, the registrar can easily create the "DS" record from the "DNSKEY" record and pass the DS record up to the registry.

However, if the authoritative servers for a domain are not with the registrar, then the registrar needs to provide some mechanism to accept a DS record to pass that up to the registry. Typically this is done through a web interface.

An issue is that many registrar web interfaces only allow the input of DS records using a listed set of DNSSEC algorithms. Any new cryptographic algorithms need to be added to the web interface in order to be accepted into the registrar’s system.

Additionally, in a manner similar to registries, many registrars perform some level of verification on the DS record to ensure it was entered "correctly". To do this verification, the registrar’s software needs to understand the algorithm used in the DS record. This requires the software to be updated to support the new algorithm.

Note that work has been standardized in [RFC8078] to provide an automated mechanism to update the DS records with a registry. If this method becomes widely adopted, registrar web interfaces may no longer be needed.

2.6. DNS Hosting Operators

DNS hosting operators are entities that are operating the authoritative DNS servers for domains and with DNSSEC are also providing the signing of zones. In many cases they may also be the registrar for domain names, but in other cases they are a separate entity providing DNS services to customers.

DNS hosting operators need to update their authoritative DNS server software to understand new cryptographic algorithms, but they also need to update their web interfaces and provisioning software to allow configuration and support of new algorithms.

2.7. Applications

Beyond the recursive resolvers, authoritative servers, web interfaces and provisioning software, it has been observed that some applications (or "apps"), particularly in the mobile environment, are including their own DNS resolvers within the app itself. These recursive resolvers are used by the app instead of the recursive resolver included with the underlying operating system. These applications that perform DNSSEC validation would need to also be updated to understand a new algorithm.

In many cases, it may be that an underlying developer library needs to be updated first. It will then depend upon how long it takes the application developer to pull in the updated library.

Outside of applications, these developer libraries are also typically used by recursive resolver software and signing software.

3. Conclusion

This document provides a view into the steps necessary for the deployment of new cryptographic algorithms in DNSSEC at the time of this publication. In order to more rapidly roll out new DNSSEC algorithms, these steps must be understood and hopefully improved over time.

It should be noted that a common theme to emerge from all discussions is a general reluctance to update or change any DNS-related software. "If it isn’t broken, don’t fix it" is a common refrain. While

perhaps understandable from a stability point of view, this attitude creates a challenge for deploying new algorithms.

One potential idea suggested during discussions was for some kind of web-based testing tool that could assist people in understanding what algorithms are supported by different servers and sites.

It is also quite clear that any deployment of new algorithms for DNSSEC use will take a few years to propagate throughout the infrastructure. This needs to be factored in as new algorithms are proposed.

4. IANA Considerations

This document does not make any requests of IANA.

5. Security Considerations

No new security considerations are created by this document.

It should be noted that there are security considerations regarding changing DNSSEC algorithms that are mentioned in both [RFC6781] and [RFC7583].

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <https://www.rfc-editor.org/info/rfc4034>.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, DOI 10.17487/RFC4035, March 2005, <https://www.rfc-editor.org/info/rfc4035>.

6.2. Informative References

[RFC3658] Gudmundsson, O., "Delegation Signer (DS) Resource Record (RR)", RFC 3658, DOI 10.17487/RFC3658, December 2003, <https://www.rfc-editor.org/info/rfc3658>.

[RFC6014] Hoffman, P., "Cryptographic Algorithm Identifier Allocation for DNSSEC", RFC 6014, DOI 10.17487/RFC6014, November 2010, <https://www.rfc-editor.org/info/rfc6014>.

[RFC6605] Hoffman, P. and W. Wijngaards, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", RFC 6605, DOI 10.17487/RFC6605, April 2012, <https://www.rfc-editor.org/info/rfc6605>.

[RFC6781] Kolkman, O., Mekking, W., and R. Gieben, "DNSSEC Operational Practices, Version 2", RFC 6781, DOI 10.17487/RFC6781, December 2012, <https://www.rfc-editor.org/info/rfc6781>.

[RFC6944] Rose, S., "Applicability Statement: DNS Security (DNSSEC) DNSKEY Algorithm Implementation Status", RFC 6944, DOI 10.17487/RFC6944, April 2013, <https://www.rfc-editor.org/info/rfc6944>.

[RFC7583] Morris, S., Ihren, J., Dickinson, J., and W. Mekking, "DNSSEC Key Rollover Timing Considerations", RFC 7583, DOI 10.17487/RFC7583, October 2015, <https://www.rfc-editor.org/info/rfc7583>.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <https://www.rfc-editor.org/info/rfc7696>.

[RFC8078] Gudmundsson, O. and P. Wouters, "Managing DS Records from the Parent via CDS/CDNSKEY", RFC 8078, DOI 10.17487/RFC8078, March 2017, <https://www.rfc-editor.org/info/rfc8078>.

[RFC8080] Sury, O. and R. Edmonds, "Edwards-Curve Digital Security Algorithm (EdDSA) for DNSSEC", RFC 8080, DOI 10.17487/RFC8080, February 2017, <https://www.rfc-editor.org/info/rfc8080>.

Appendix A. Acknowledgements

The information in this document evolved out of several mailing list discussions and also through engagement with participants in the following sessions or events:

- DNSSEC Workshop at ICANN 53 (Buenos Aires)
- DNSSEC Workshop at ICANN 55 (Marrakech)
- Spring 2016 DNS-OARC meeeting (Buenos Aires)
- various IETF 95 working groups (Buenos Aires)
- Panel session at RIPE 72 (Copenhagen)
- DNSSEC Workshop at ICANN 56 (Helsinki)

The authors thank the participants of the various sessions for their feedback.

Appendix B. Changes

NOTE TO RFC EDITOR - Please remove this "Changes" section prior to publication. Thank you.

- Revision -06 added in references to RFCs 8080 and 8078 and updated Ondrej Sury's affiliation to ISC.
- Revision -05 corrected typos around two other references that did not appear in -04.
- Revision -04 corrected the references which did not appear in -03 due to an error in the markdown source.
- Revision -03 removed the reference to the location of the ISP in the text added in version -02.
- Revision -02 added text to the resolver section about an example where resolver software did not correctly follow RFC 4035 and treat packets with unknown algorithms as unsigned. The markdown source of this I-D was also migrated to the markdown syntax favored by the 'mmark' tool.
- Revision -01 adds text about authoritative servers needing an update if the algorithm is for NSEC/NSEC3. Also expands acknowledgements.

Authors' Addresses

Dan York
Internet Society

Email: york@isoc.org
URI: https://www.internetsociety.org/

Ondrej Sury
ISC

Email: ondrej@isc.org

Paul Wouters
Red Hat

Email: pwouters@redhat.com

Olafur Gudmundsson
CloudFlare

Email: olafur+ietf@cloudflare.com